PATHS IN INFINITE TREES: LOGICS AND AUTOMATA

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades einer Doktorin der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatikerin

ALEXANDRA SPELTEN

aus Koblenz, Rheinland-Pfalz

Berichter: Universitätsprofessor Dr. Dr.h.c. Wolfgang Thomas
Universitätsprofessor Dr. Erich Grädel

Tag der mündlichen Prüfung: 28. Januar 2013

Abstract

In this thesis, several logical systems over infinite trees and infinite words are studied in their relation to finite automata.

The first part addresses (over the infinite binary tree) "path logic" and "chain logic" as fragments of monadic second-order logic that allow quantification over paths, respectively subsets of paths of the binary tree. Many systems of branching-time logic are subsumed by chain logic. We introduce *ranked alternating tree automata* as a computation model that characterizes chain logic. The main idea is to associate ranks to states such that in an automaton run, starting from the root, the ranks have to decrease and are allowed to remain unchanged only in one direction (either in existential or universal branching).

The second part of the thesis is motivated by chain logic over infinite-branching trees (where the successors of a node are indexed by natural numbers). A path through the $\mathbb{N}$-branching tree is given by an ω-word over the infinite alphabet $\mathbb{N}$. As a preparation for the study of path logics over such trees, we develop a theory of logics and automata over infinite alphabets, more precisely over *alphabet frames* $(\mathcal{M}, \mathcal{L})$, given by a relational structure $\mathcal{M}$, supplying the alphabet, and a logic $\mathcal{L}$ that is used in specifying letter properties and automaton transitions. Two types of automata (and logics) for the specification of word properties are presented, depending whether or not relations between successive letters are included. We obtain results that clarify under which circumstances the nonemptiness problem is solvable, and we apply these results to show (un-)decidability results on path logics over infinitely-branching trees that result from given structures by weak and strong "tree iteration".

Zusammenfassung

In der vorliegenden Arbeit werden verschiedene Logiksysteme über unendlichen Bäumen und unendlichen Wörtern in ihrer Beziehung zu endlichen Automaten untersucht.

Dazu analysieren wir im ersten Teil als Fragmente von monadischer Logik zweiter Stufe die Logiken "Pfadlogik" und "Kettenlogik" (über dem unendlichen Binärbaum), die Quantifizierung über Pfaden bzw. Teilmengen von Pfaden des Binärbaums erlauben. Viele Systeme der "branching time logic" werden von Kettenlogik erfaßt. Als automatentheoretische Charakterisierung von Kettenlogik führen wir *alternierende Rang-Baumautomaten* ein, deren Zuständen jeweils ein Rang zugeordnet werden. Die Ketten werden dann simuliert, indem sichergestellt wird, dass in einem Lauf des Automaten, beginnend in der Wurzel, die Ränge stetig abfallen müssen und nur in höchstens einer Richtung des Baumes erhalten bleiben dürfen (in existentieller oder universeller Verzweigung).

Der zweite Teil dieser Dissertation wird von Kettenlogik über unendlich verzweigten Bäumen (wobei die Nachfolger eines Knotens durch natürliche Zahlen indiziert werden) motiviert. Ein Pfad durch einen solchen $\mathbb{N}$-verzweigten Baum wird durch ein ω-Wort über dem unendlichen Alphabet $\mathbb{N}$ beschrieben. Als Vorbereitung für die Studie von Pfadlogiken über solchen Bäumen entwickeln wir eine Theorie von Logiken und Automaten über unendlichen Alphabeten; genauer über *Alphabetrahmen* $(\mathcal{M}, \mathcal{L})$, welche durch eine Struktur $\mathcal{M}$, die das Alphabet stellt, und eine Logik $\mathcal{L}$, mit der Eigenschaften von Buchstaben und Automatentransitionen spezifiziert werden, gegeben sind. Wir präsentieren zwei Typen von Automaten (und Logiken) für die Spezifikation von Worteigenschaften, abhängig davon, ob Relationen aufeinanderfolgender Buchstaben enthalten sind oder nicht. Wir erhalten Resultate, die klarstellen, unter welchen Bedingungen das Leerheitsproblem lösbar ist, und wenden diese daraufhin an, um (Un-) Entscheidbarkeitsergebnisse für Pfadlogiken über unendlich verzweigten Bäumen zu zeigen, welche aus gegebenen Strukturen durch schwache und starke "Baumiteration" entstehen.

Contents

Chapter 1

Introduction

Many fields of theoretical computer science that deal with verification or synthesis of state-based systems have their foundation in the theory of finite automata, more precisely in results that connect automata with logical systems. The first results of this kind were shown in the 1960s, first by Büchi, Elgot, and Trakhtenbrot connecting finite automata and monadic second-order logic over finite words as formalisms that are expressively equivalent (cf. [Büc60, Elg61, Tra61]). Further steps were analogous results over infinite words (Büchi [Büc62]), finite trees (Doner [Don70], Thatcher, Wright [TW68]), and finally the celebrated "tree theorem" of Rabin [Rab69] that showed the expressive equivalence between "Rabin tree automata" and monadic second-order logic (MSO-logic) over the binary tree. As a main application, the MSO-theory of the infinite binary tree was shown to be decidable.

The use of MSO-logic can be motivated by the aim to describe runs of finite automata on the structure under consideration. A run defines a "coloring" (by states) on this input structure, and thus the existence of a run (or a finite coloring, say with k colors) can be expressed by a statement on the existence of k subsets of the structure's domain, one for each color, collecting those elements that carry this color. Thus monadic logic is a natural choice to describe existence of runs, explaining partly the "match made in heaven" [Var03] between MSO-logic and finite automata.

However, in the applications of MSO-logic (over trees) in the areas of verification and synthesis, certain fragments of MSO-logic are taken rather than full MSO-logic. This is visible, for example, in the systems CTL and CTL* of "computation tree logic". Here only quantifiers over paths are needed (besides first-order quantifiers), i.e., very special subsets of the tree. For example the CTL*-formula

$$p_1 \wedge E(Gp_2 \wedge FEFp_3)$$

says

> "p_1 is true in s_0, and starting from s_0 there is an infinite path π through the model such that all states on π satisfy p_2, and in some state of π a path π' starts with some state satisfying p_3".

The situation does not change when adding "automaton operators" on the paths, leading to the system ECTL* of "extended CTL*". Here a further kind of quantifier enters, namely quantifiers over *subsets of paths*, also called *chains*.

In papers of the 1980s, Thomas [Tho84, Tho87] has introduced and studied these subsystems of MSO-logic, called *path logic* and *chain logic*. It was shown that they are strictly less expressive than MSO-logic (and that path logic is strictly weaker than chain logic). This weakness was then exploited to show the decidability of the chain logic theory of an expanded tree structure, namely the binary tree expanded by the "equal-level predicate" (connecting two tree nodes if they occur on the same level of the tree, i.e., have the same distance to the root).

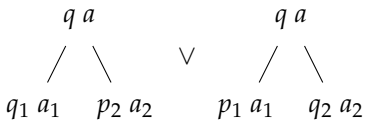
Also a kind of "dual approach" to subsystems of MSO-logic was pursued in the consideration of *antichain logic* (where second-order quantifiers range over sets whose elements are pairwise incomparable with respect to partial tree ordering, cf. [Tho84]). In these studies, the "composition method" adapted to path logic and chain logic was applied. This method is built on the notion of "types" that capture the equivalence between structures with respect to formulas of a given quantifier-rank, and results on the computation of types of structures that are composed from substructures (e.g. words composed by the concatenation of segments).

A missing piece in the picture was the reference to an automaton model that captures chain logic, resp. path logic. The first objective of this thesis is to provide such a model. A second objective is to study path and chain logic over infinitely branching trees, taking an approach which first addresses ω-automata over infinite alphabets.

Outline

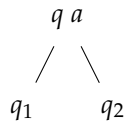
The development of a model of tree automaton for path and chain logic takes place as follows: In the first part of this thesis, we introduce "ranked alternating tree automata" with specific properties in their transitions, and we show their expressive equivalence to chain logic.

Let us sketch this model of automaton in intuitive words. The easiest situation is given by the "implementation" of an existential chain quantifier, say in a formula $\exists X \varphi(X)$. Here the automaton guesses a path and a subset of this path, and checks the rest of the tree for the satisfaction of the formula φ. Such a test of the tree naturally suggests to associate a higher rank to states on the guessed path than on the rest of the tree. Formally, this idea is captured by the requirement that we have transitions of the form

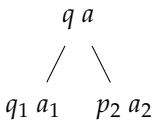


where the rank of q is preserved in q_1 (the case that the path branches left) and in q_2 (the case that the path branches right) but that p_1, p_2 have to have a lower rank. In order to include universal quantification, we allow universal states where in the applicable transition $\wedge$ appears in place of $\vee$.

The model of ranked alternating tree automaton shows transitions of a form that slightly differs from the format



used in classical tree automata theory. The present version involves an "overlapping" of neighboring transitions in a run, since e.g. on the left child of a transition



the use of q_1 is already prescribed by the displayed parent transition. This modification is convenient in order to treat "guessing a direction of a path" and an association of ranks in a consistent way. Transitions that involve overlapping are well-known from automata over two dimensional words (pictures), see e.g. [GRST96].

The second part of the thesis deals with logics and automata motivated by *infinitely branching trees*. For this we recall the main idea of [Tho92] that shows the decidability of the chain logic theory of the binary tree expanded by the equal-level predicate: Each chain can be represented by a pair of ω-words – the first describing a path as a sequence of symbols "left", "right" (or 0, 1), the second describing a subset of this path (hence also a 0-1-sequence). In this way a reduction of the chain theory of the binary tree to Büchi's logic S1S (or the MSO-theory of $(\mathbb{N}, +1)$) is achieved. This coding needs a generalization when the infinitely branching tree is studied; in this case we need MSO-logic over infinite words built up from the alphabet $\mathbb{N}$.

Thus, to cope with infinite branching, we develop a theory of automaton-definable ω-languages over infinite alphabets, extending work of [Bès08]. Let us emphasize two aspects.

An infinite alphabet does not arise as a "set of symbols" but comes as a mathematical structure $\mathcal{M}$ (whose universe is denoted M) together with a logic $\mathcal{L}$ that allows to specify properties of elements of M (or properties of elements of n-tuples from M^n). In the work of Bès one finds the structure $\mathcal{M} = (\mathbb{N}, +, 0)$ and $\mathcal{L} =$ first-order logic (FO-logic) as the leading example. In our set-up, we work

out the results of Bès in a general framework, using a pair $(\mathcal{M}, \mathcal{L})$ as "alphabet frame". In order to obtain effective results in close analogy to the classical theory of regular languages over finite alphabets, in particular on the equivalence between MSO and automata, the assumption suffices that the $\mathcal{L}$-theory of $\mathcal{M}$ is decidable. Our results are not surprising but may be useful in a more systematic study of infinite alphabets. In particular, it is necessary to clearly separate the logic $\mathcal{L}$ that allows to define letter-properties from the logic (for example MSO or FO) that allows to define word-properties. We thus arrive at a notation and terminology that is necessarily complex, for example when speaking of MSO-definable languages over the alphabet frame $(\mathcal{M}, \mathcal{L})$, and showing the equivalence of MSO-logic and finite automata (we restrict ourselves here to the case of languages of finite words and ω-words).

The second aspect is the study of extended frameworks in which the main weakness of the above mentioned approach is remedied, namely the inability of MSO-logic and of automata over an alphabet frame to specify interesting relations between successive letters. As a simple example, we note that the two-letter-word language consisting of the words aa where $a \in M$ is not definable. This problem also arises in the investigation of tree models obtained from a given structure $\mathcal{M}$ (with universe M). The natural model of "tree iteration of M" has words of M^* as its elements and the prefix relation as the partial tree ordering. In the set-up of "weak tree iterations", proposed by Shelah [She75] and Stupp [Stu75] there are no means to recognize an element waa as one where the two last steps of child-formation are the same. This weakness is overcome in the "strong tree iteration" proposed by Muchnik [Sem84] (and further studied by Walukiewicz [Wal02] and Courcelle and Walukiewicz [CW98]). Here the "clone predicate" is added to the tree structure, defined to be true of a tree element wa if w has the form ua.

In the second section of the second part of this thesis we concentrate on *paths* in such tree structures, and hence we study words and ω-words in a logical and automata-theoretic framework where possibilities to compare successive letters are offered.

In fact, there are different options to introduce such possibilities. We pursue here a concept of "strong automata" over words whose successive input letters are linked in the automaton transitions. Such a transition from state p to state q can be taken when a letter $\overline{k} \in M^n$ that was used to reach state p and the next letter $\overline{a} \in M^n$ to reach state q satisfy a "transition condition" $\varphi_{pq}(\overline{y}, \overline{x})$, i.e, for the alphabet structure we have

$$\mathcal{M} \models \varphi_{pq}[\overline{k}, \overline{a}].$$

We speak of "strong automata" in this context and study their expressiveness and decidability properties.

A first result says that even for the alphabet frame $(\mathcal{M}, \mathcal{L})$ with $\mathcal{M} = (\mathbb{N}, +1)$

and $\mathcal{L} =$ FO-logic, the nonemptiness problem for strong automata on words over M^2 is undecidable (proved by a simple coding of 2-register machines). Thus we have to concentrate on words over alphabets M^n with $n = 1$. Here we show that an effective theory of automata is possible when the MSO-theory of the alphabet structure $\mathcal{M}$ is decidable. Thus for the case of strong automata we need (not surprisingly) more assumptions than before for the "weak" automata.

In the final section we return to the domain of tree models and apply the obtained results to constructions of tree iterations, extending known results on chain logic from finitely branching to infinitely branching trees. The background of our study is the above-mentioned result of Thomas [Tho87] that the chain logic theory of the binary tree is decidable even when the signature of the tree is extended by the "equal-level predicate". We investigate this result in two kinds of tree structures which are in general infinitely branching: the "weak tree iteration" $\mathcal{M}^\#$ of a structure $\mathcal{M}$ (in the sense of Shelah [She75] and Stupp [Stu75]) and the "strong tree iteration" $\mathcal{M}^*$ (in the sense of Muchnik [Sem84]). We consider chain logic over such tree structures and ask whether the "equal-level predicate" can again be added while giving a decidable theory. For the weak tree iteration, we show a positive result: If (for some given logic $\mathcal{L}$) the $\mathcal{L}$-theory of $\mathcal{M}$ is decidable, so is the chain logic theory of $\mathcal{M}^\#$, extended by the equal-level predicate and by built-in $\mathcal{L}$-formulas restricted to tree levels. This seems to be an interesting extension of Thomas' result [Tho87] on the binary tree. However, for the strong tree iteration we obtain a radically different situation when passing from finitely branching to infinitely branching trees: We show that for $\mathcal{M} = (\mathbb{N}, +1)$ the chain logic theory of $\mathcal{M}^*$ extended by the equal-level predicate is undecidable. Also we show the following result for the domain of finitely branching trees: The binary tree extended by the equal-level predicate has an undecidable theory in the system of chain logic when it is enhanced by MSO-quantifiers that range over given tree levels. Thus the combination of quantification over "vertical" and "horizontal" sets in a binary tree (which are chains, resp. subsets of given levels) yields a more complex situation than full monadic second-order quantification without the equal-level predicate. The latter determines a decidable theory by Rabin's tree theorem.

The thesis concludes with the sketch of three research directions that can be pursued starting from the present work.

Acknowledgments

First of all, I would like to thank my supervisor, Wolfgang Thomas, for giving me the opportunity to pursue the path of scientific research. He has suggested the topic for this thesis, and his guidance has been of great value to my work on this thesis. I would like to express my gratitude to Christof Löding for many insightful discussions and his careful proofreading. His substantial advice fixed the format of the automaton model in Chapter 3.

I would also like to thank Erich Grädel for his kind (and prompt) readiness to review this thesis as a co-examiner.

Thanks go also to Jörg Olschewski for some helpful discussions, and of course to Nanne Zwart for his love and support (and considerable patience, too).

Chapter 2

Background on Monadic Second-Order Logic and Automata

In this chapter, the notations, basic definitions, and terminology are formalized that were used informally in the introduction. Standard notation is employed as introduced e.g. in [HU79].

Words and ω-words

For a nonempty set M of *symbols*, let M^* denote the set of all finite sequences (*words*) over M, and M^+ the set of all nonempty finite sequences over M. For $w \in M^*$, say $w = a_0 \cdots a_{n-1}$ for some $n \in \mathbb{N}$ and $a_i \in M$ for all $0 \leq i \leq n-1$, let $|w| := n$ denote the *length* of w, and $w(i)$ denote the i-th letter of w, i.e. $w(i) = a_i$. The *empty word* is denoted by ε with $|\varepsilon| = 0$. A *language* over a set M is a subset of M^*.

The *concatenation* of words u and v, denoted by $u \cdot v$ or shortly uv, is obtained by appending v to u, i.e. uv is the word $u(1) \cdots u(|u|)v(1) \cdots v(|v|)$. Moreover, the concatenation of two languages K and L is defined by $KL := \{uv \mid u \in K, v \in L\}$. For a language L, define $L^0 := \{\varepsilon\}$, $L^{i+1} := L^i \cdot L$, and $L^* := \bigcup_{i \geq 0} L^i$. L^* is called the *Kleene closure* of L. The set $RE(M)$ of *regular expressions* is built up inductively from $\emptyset$, ε, and all symbols $a \in M$ by application of concatenation, Kleene closure, and union; their semantics is defined in the standard way. We write $L(r)$ for the language denoted by the expression r.

The word $u \in M^*$ is called a *prefix* of $w \in M^*$, if there is a word $v \in M^*$ such that $w = u \cdot v$; we write $u \preceq w$. A language $L \subseteq M^*$ is *prefix closed* iff for each $w \in L$, it holds that every prefix of w is also in L.

In the context of logic, the word $w = a_0 \cdots a_{n-1}$ is represented by the relational structure

$$\underline{w} = (dom(w), Suc^w, <, (P_a^w)_{a \in M})$$

called the *word model* for w, where $dom(w) = \{0, \ldots, n-1\}$ is the set of (letter) positions of w (the "domain" of w), Suc^w is the successor relation on $dom(w)$ with $(i, i+1) \in Suc^w$ for $0 \leq i \leq n-1$, $<$ is the natural order on $dom(w)$, and the P_a^w are unary predicates, collecting for each label a the letter positions of w which carry a, thus $P_a^w = \{i \in dom(w) \mid a_i = a\}$. A word model $\underline{w}$ can be viewed as a vertex labeled graph with edge relation Suc^w (which induces the linear ordering $<$).

This framework is easily adapted to ω-words over a given alphabet M, i.e., to sequences $\alpha = a_0a_1a_2\cdots$ with $a_i \in M$. The corresponding structures $\underline{\alpha}$ are of the form

$$\underline{\alpha} = (\omega, Suc^\alpha, <^\alpha, (P_a^\alpha)_{a\in M})$$

where the domain is fixed as the set $\omega = \{0, 1, 2, \ldots\}$ of natural numbers; again we have $P_a^\omega = \{i \in \mathbb{N} \mid a_i = a\}$.

Trees

We will mostly restrict ourselves to proper binary trees, in which each node is either a leaf or has two successors, which are ordered as left and right successor. This eases notation but covers most typical features arising with trees. Thus, nodes of trees will be represented as finite words over the alphabet $\{0, 1\}$ (where 0 denotes a branch to the left while 1 denotes a branch to the right), and tree domains will be prefix closed subsets D of $\{0, 1\}^*$, such that for any word $w \in D$ either both or none of $w0$, $w1$ also belong to D (reflecting the case that w is an inner node resp. a leaf of a tree). A tree τ over alphabet M is presented as a map $\tau : dom(\tau) \to M$ where $dom(\tau)$ is a tree domain. Graphically we present trees in the usual form (cf. e.g. Figure 2.1(b)). The corresponding relational structure has the form

$$\underline{\tau} = (dom(\tau), Suc_0^\tau, Suc_1^\tau, <, (P_a^\tau)_{a\in M}).$$

Here, Suc_0^τ, Suc_1^τ are the left respectively right successor relations over $dom(\tau)$ (with $(u, u0) \in Suc_0^\tau$ and $(u, u1) \in Suc_1^\tau$ for $u, u0, u1 \in dom(\tau)$), $<^\tau$ is the proper prefix relation over $dom(\tau)$, and $P_a^\tau = \{u \in dom(\tau) \mid \tau(u) = a\}$. We say that a tree is finite if its domain is finite; as infinite trees over M we shall consider only the *full* binary trees, i.e., maps from $\{0, 1\}^*$ to M. We denote by T_M the set of all finite trees over M, and by T_M^ω the set of infinite (full binary) trees over M. A path π through τ is a maximal subset of $dom(\tau)$ linearly ordered by $<$. The subtree $\tau_{\downarrow u}$ of τ at node $u \in dom(\tau)$ is given by $dom(\tau_{\downarrow u}) = \{v \in \{0, 1\}^* \mid uv \in dom(\tau)\}$ and $\tau_{\downarrow u}(v) = \tau(uv)$ for $v \in dom(\tau_{\downarrow u})$. An infinite tree $\tau \in T_M^\omega$ is said to be regular if there are only finitely many distinct subtrees $\tau_{\downarrow u}$ (where $u \in \{0, 1\}^*$).

Besides binary trees, we will sometimes generalize to trees over a *ranked alphabet*. A ranked alphabet is a nonempty finite set M of symbols where each symbol $a \in M$ is assigned a *rank* or arity $rk(a) \in \mathbb{N}$. In a tree τ over a ranked alphabet, each a-labeled node has exactly $rk(a)$-many successors. In this thesis, we will call a tree τ "finitely branching" if the branching degree of τ is bounded.

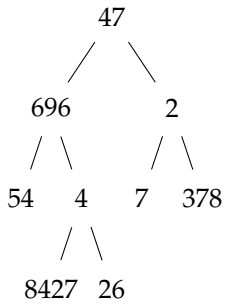
Example 2.1. Let the domain M be the natural numbers, i.e., $M = \mathbb{N}$. Then the finite word $w \in M^*$ with

$$w = \underset{a_0}{1}\ \underset{a_1}{13}\ \underset{a_2}{24}\ \underset{a_3}{599}\ \underset{a_4}{2}\ \underset{a_5}{1684}\ \underset{a_6}{13}\ \underset{a_7}{12}$$

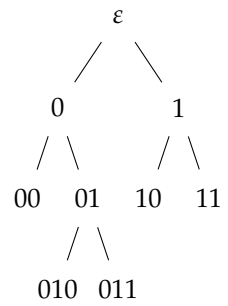
Figure 2.1 Word w and tree τ from Example 2.1.

$$w = 1 \quad 13 \quad 24 \quad 599 \quad 2 \quad 1684 \quad 13 \quad 12$$

(a) Word w.



(b) Tree τ.



(c) Domain $dom(\tau)$ of τ.

with $dom(w) = \{0, \ldots, 7\}$ is depicted in Figure 2.1(a). In the word model $\underline{w}$, the unary predicate for letter 13 is $P_{13}^{w} = \{1, 6\}$. An infinite word $\alpha \in M^{\omega}$ is e.g.

$$\alpha = 27 \quad 54 \quad 8 \quad 910 \quad 462 \quad 3 \quad 3 \quad 3 \quad \ldots$$

Let τ be a finite binary tree over $M = \mathbb{N}$ with domain $dom(\tau) = \{\varepsilon, 0, 1, 00, 01, 10, 11, 010, 011\}$. The graphical notation as well as the domain of such a tree τ are depicted in Figure 2.1. ◀

In Chapter 4 we use a framework in which arbitrary relational structures appear. These structures are of the form $\mathcal{M} = (M, R_1^M, \ldots, R_k^M)$ where each R_i^M is a relation over M of any arity ≥ 1.

Logics

Properties of words or trees can be formalized in logical languages. We will first introduce here first-order logic FO and monadic second-order logic MSO.

Consider word models over the alphabet M. The corresponding first-order language has variables $s, t, \ldots$ ranging over positions in word models, and is built up from atomic formulas of the form

$$s = t, \; Suc(s, t), \; s < t, \; P_a(s) \text{ for } a \in M$$

by means of the Boolean connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, and the quantifiers $\exists$ and $\forall$. We call the set of employed relation symbols Suc, $<$, P_a the *signature* of this first-order language (equality $=$ is always assumed to be available). The notation $\varphi(s_1, \ldots, s_n)$ indicates that at most the variables $s_1, \ldots, s_n$ occur free, i.e., not in

the scope of a quantifier, in the formula φ. A *sentence* is a formula without free variables.

We will employ some standard results of quantifier logic, especially the *prenex normal form* into which each first-order formula can be transformed (cf. [EFT07]). In this form, a prefix of quantifiers preceeds a quantifier-free kernel.

We write $(\underline{w}, p_1, \ldots, p_n) \models \varphi(s_1, \ldots, s_n)$ if $p_1, \ldots, p_n$ are positions from $dom(w)$ and φ is satisfied in $\underline{w}$ when $=$, Suc, $<$, P_a are interpreted by equality, Suc^w, $<^w$, P_a^w, respectively, and $p_1, \ldots, p_n$ serve as interpretations of $s_1, \ldots, s_n$, respectively. The empty model is usually excluded from the framework of mathematical logic to avoid having to handle special cases. Hereafter, we will allow the empty word ε as a member of formal languages, and admit the empty model as interpretation of sentences. We will stick to the natural convention that $\underline{\varepsilon}$ satisfies universal sentences $\forall s \varphi(s)$ but does not satisfy existential sentences $\exists s \varphi(s)$.

The language defined by the sentence φ is denoted as $L(\varphi) = \{w \in M^* \mid \underline{w} \models \varphi\}$. Similarly, the ω-language defined by φ is $L_\omega(\varphi) = \{\alpha \in M^\omega \mid \underline{\alpha} \models \varphi\}$.

If a first-order sentence φ as above exists with $L = L(\varphi)$, resp. with $L = L_\omega(\varphi)$, we call a language $L \subseteq M^*$, resp. an omega-language $L \subseteq M^\omega$ FO$[Suc, <]$-definable, or simply FO-definable.

Analogously, first-order formulas over tree models are introduced. The signature is adapted accordingly, along with the interpretation of its relation symbols. Thus, for (binary) trees we employ the relation symbols Suc_0, Suc_1 for the two successor relations, and $<$ now stands for the partial order of the proper prefix relation over tree domains. We denote the set of finite, resp. infinite trees (over a given alphabet M) which satisfy the sentence φ by $T(\varphi)$, resp. $T_\omega(\varphi)$.

Sometimes, it may be more convenient to use function symbols rather than relation symbols, for example the symbol Succ for successor function instead of the relation Suc as introduced above. This allows shorter formalizations especially if compositions of functions are considered. For instance, we can write $t = \mathrm{Succ}(\mathrm{Succ}(\mathrm{Succ}(s)))$ or $t = \mathrm{Succ}^3(s)$ instead of $\exists t_1 \exists t_2 (Suc(s, t_1) \wedge Suc(t_1, t_2) \wedge Suc(t_2, t))$. Since we can always eliminate function symbols in terms of relation symbols, we will restrict our general considerations to the relational case, although we may use the functional notation to ease readability where appropriate.

We now extend the logical formalism by second-order variables $S, T, \ldots$ which range over sets of elements of model, i.e., sets of letter positions, sets of tree nodes, or sets of graph vertices. We also introduce the corresponding atomic formulas $S(s), S(t), \ldots$, with the intended meaning "s belongs to the set S", "t belongs to the set S", etc. With sets being "monadic second-order objects" (in contrast to relations of higher arity, which are "polyadic"), the resulting system is called *monadic second-order logic* or short: MSO-logic.

Again, for second-order formulas a prenex normal form exists where the second-order quantifiers may be shifted in front of all first-order quantifiers [Büc62]. Note that in MSO-logic the order relation $<$ over words becomes definable in terms of the successor relation Suc: Over words models, $s < t$ is equivalent to $\neg s = t \wedge \forall S\big(\big(S(s) \wedge \forall t' \forall t''(S(t') \wedge Suc(t', t'') \rightarrow S(t'')\big)\big) \rightarrow S(t)\big)$. Thus we obtain that any FO[$<$]-definable word language is also MSO[Suc]-definable, or simply "MSO-definable". Over trees, a similar definition of the partial tree order $<$ (which is the proper prefix relation over the tree domain $dom(\tau)$ for a given tree τ) can be given in terms of the two successor relations Suc_0 and Suc_1.

In the study of monadic second-order logic, it will be useful to consider a modified logical system of the same expressive power, called MSO_0-logic. It has a simpler syntax in which first-order variables are obsolete: The idea is to simulate first-order variables by singleton sets. Thus, $\{s\} \subseteq S$ will replace $S(s)$, etc. We therewith obtain new atomic formulas for MSO_0-logic, namely (with given alphabet M):

$$S \subseteq T,\ \mathrm{Sing}(S),\ Suc(S, T),\ S \subseteq P_a \text{ for } a \in M$$

with the meaning that S is a subset of T, S is a singleton, S and T are singletons $\{s\}$ and $\{t\}$ with $Suc(s, t)$, and S is a subset of P_a, respectively.

One can easily give a translation from MSO-logic to MSO_0-logic via an induction over the construction of MSO-formulas. For instance,

$$\forall s(P_a(s) \rightarrow \exists t(Suc(s, t) \wedge U(t)))$$

can be rewritten as

$$\forall S(\mathrm{Sing}(S) \wedge S \subseteq P_a \rightarrow \exists T(\mathrm{Sing}(T) \wedge Suc(S, T) \wedge T \subseteq U)).$$

Over trees, we would use formulas $Suc_i(S, T)$ instead of $Suc(S, T)$ for $i \in \{0, 1\}$.

An MSO-formula $\varphi(S_1, \ldots, S_n)$ with at most the free variables $S_1, \ldots, S_n$ is interpreted in a word model, resp. tree model, with n designated subsets $P_1, \ldots, P_n$. Such a model represents a word, resp. tree, over the expanded alphabet $M' = M \times \{0, 1\}^n$, where the label $(a, b_1, \ldots, b_n)$ of position p, resp. node p, indicates that p carries the label $a \in M$ and that p belongs to P_i iff $b_i = 1$. For example, the ω-word model $(\underline{\alpha}, P_1, P_2)$ where $\alpha = babbaaa\ldots$, P_1 is the set of prime numbers, P_2 is the set of odd numbers, will be identified with the following ω-word over $\{a, b\} \times \{0, 1\}^2$, where the letters are written as columns:

α	b	a	b	b	a	a	a	a	a	a	
P_1	0	0	1	1	0	1	0	1	0	0	$\cdots$
P_2	0	1	0	1	0	1	0	1	0	1	

In the following, we will often identify the set expansion of a model with a model over an extended alphabet.

Note that in the logical framework there is no essential difficulty in transferring definability notions from the domain of words to the more extended domain of trees, and that also the transition from finite models to infinite models does not involve any conceptual problems. It is only necessary to adapt the signature under consideration and to change the class of admitted models.

In Chapter 4 we shall refer to arbitrary logics $\mathcal{L}$ rather than concrete logics such as FO-logic or MSO-logic. The notion of logic is then to be understood in the field of abstract model theory (cf. [BF85, EFT07]) with the appropriate conditions. For example, when referring to the assumption that the "$\mathcal{L}$-theory of a structure $\mathcal{M}$ is decidable" we implicitly assume that the syntax of $\mathcal{L}$ is effective.

Automata on Words and Trees

Let us recall regular languages, i.e., languages that are recognized by finite automata (cf. e.g. [HU79]).

A *nondeterministic finite automaton (NFA)* on words over a set M is of the form $\mathcal{A} = (P, M, p_0, \Delta, F)$ with

- nonempty finite state set P,
- finite input alphabet M,
- initial state $p_0 \in P$,
- transition relation $\Delta \subseteq P \times M \times P$, and
- set of final states $F \subseteq P$.

A *run* of $\mathcal{A}$ from $p \in P$ via a word $w = w(1) \cdots w(|w|)$ to $p' \in P$ is a sequence $\rho = \tilde{p}_0, \ldots, \tilde{p}_{|w|}$ of states with $\tilde{p}_0 = p$, $(\tilde{p}_i, w(i+1), \tilde{p}_{i+1}) \in \Delta$ for $0 \leq i < |w|$, and $\tilde{p}_{|w|} = p'$. If such a run exists, it is denoted by $\mathcal{A} : p \xrightarrow{w} p'$. $\mathcal{A}$ *accepts* w iff $\mathcal{A} : p_0 \xrightarrow{w} p'$ for some $p' \in F$. The language recognized by $\mathcal{A}$ is denoted by $L(\mathcal{A}) = \{w \in M^* \mid \mathcal{A} \text{ accepts } w\}$. The class of languages recognized by NFAs over M is exactly the class of languages that can be defined by regular expressions over M, which is called the class of *regular languages*.

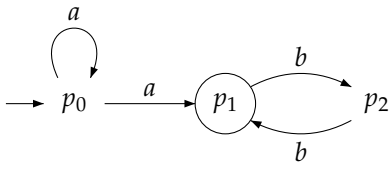
A *deterministic finite automaton (DFA)* on words over M is of the form $\mathcal{A} = (P, M, p_0, \delta, F)$ with

- P, M, p_0, and F as in the nondeterministic case, and
- transition function $\delta : P \times M \to P$.

To obtain a simpler notation, $\delta^* : P \times M^* \to P$ is defined by $\delta^*(p, w) =$ the unique state reached by $\mathcal{A}$ when starting in p and processing w. $\mathcal{A}$ accepts $w \in M^* :\Leftrightarrow \delta^*(p_0, w) \in F$. The language $L(\mathcal{A})$ is defined analogously to the nondeterministic case: $L(\mathcal{A}) = \{w \in M^* \mid \mathcal{A} \text{ accepts } w\}$.

We recall that the powerset construction allows us to reduce NFAs to DFAs, thus these types of automata are of the same expressive power.

Example 2.2. As usual, we present an NFA by a graphical picture. For example, consider the NFA $\mathcal{A} = (P, M, p_0, \Delta, F)$ over the set $M = \{a, b\}$ with state set $P = \{p_0, p_1, p_2\}$, transition relation $\Delta = \{(p_0, a, p_0), (p_0, a, p_1), (p_1, b, p_2), (p_2, b, p_1)\}$, and final state set $F = \{p_1\}$.



The language of $\mathcal{A}$ contains exactly those words over M that start with any positive number of a's which can be followed by an even number (possibly 0) of b's. ◀

Theorem 2.3 (Büchi [Büc60], Elgot [Elg61], Trakhtenbrot [Tra61]). *A language of finite words is recognizable by a finite automaton iff it is* MSO*-definable, and both conversions, from automata to formulas and vice versa, are effective.*

We give a brief sketch of the well-known proof (cf. e.g. [Tho90]) by considering both inclusions of the equivalence. As for the direction from automata to logic, a sentence φ has to express the existence of a run for a given automaton $\mathcal{A}$ such that

$$\underline{w} \models \varphi \text{ iff } w \in L(\mathcal{A}).$$

This can easily be accomplished if we take the structural properties of an automaton into account, i.e., for each state q_i of the automaton the sentence φ assumes the existence of a set S_i. Naturally the automaton can only be in one state at each point of time. Thus, we state that all stets S_i form a partition, which can easily be defined in MSO. Then it remains to ensure that the run starts in the initial state of the automaton, that at the last transition, a final state is reached, and for each successive points of time, a valid transition is applied. For the former two statements, MSO-sentences are straightforward, the latter condition simply lists all possible transitions in Δ; e.g. a transition (q_1, a, q_2) yields $(S_1(s) \wedge P_a(s) \wedge S_2(t))$ for successive s, t.

As for the direction from logic to automata, we inductively construct a finite automaton for an MSO_0-formula. As mentioned above, a formula $\varphi(S_1, \ldots, S_n)$

is interpreted in a word model over the expanded alphabet $M' = M \times \{0,1\}^n$, where each designated subset corresponds to a variable S_i. Then, the atomic MSO_0-formulas of subset relations, being a singleton, or successor relation are easily translated within the transitions of basic automata. For the induction step we exploit the fact that NFAs are closed under Boolean operations and projection, thus completing the desired construction.

The theory of finite tree automata arises as a straightforward extension of the theory of finite word automata when words are viewed as unary terms. Only a brief introduction to finite tree automata over finite binary trees is provided here as a preparation for automata over infinite trees. We thus concentrate on the model of (nondeterministic) top down tree automata; for extensive analyses cf. e.g. [Don70], [TW68], [GS84], [CDG+07].

Considering a graphical notation of a tree with the root symbol at the top (cf. e.g. Figure 2.1(b)), a *nondeterministic top down tree automaton (N↓TA)* on binary trees starts its computation at the root of the tree and moves towards the leaves. Formally, an N↓TA is a tuple $\mathcal{A} = (Q, M, q_0, \Delta, F)$ with nonempty finite state set Q, initial state $q_0 \subseteq Q$, and transition relation $\Delta \subseteq Q \times M \times Q^2$, and "final combinations" $F \subseteq Q \times M$ which assign states to the leaves of a tree. A run of $\mathcal{A}$ on a binary tree τ is a function $\rho : dom(\tau) \to Q$ such that $\rho(\varepsilon) = q_0$, if s is a leaf, then $(\rho(s), \tau(s)) \in F$, and otherwise $(\rho(s), \tau(s), \rho(s0), \rho(s1)) \in \Delta$. A tree τ is accepted by $\mathcal{A}$ if there exists a run ρ of $\mathcal{A}$ on τ such that at each leaf, we have a "final combination". The language recognized by $mathcal A$ is denoted by $T(\mathcal{A}) := \{\tau \in T_M \mid \mathcal{A} \text{ accepts } t\}$. *Regular languages of binary trees* are those that can be accepted by N↓TAs on binary trees.

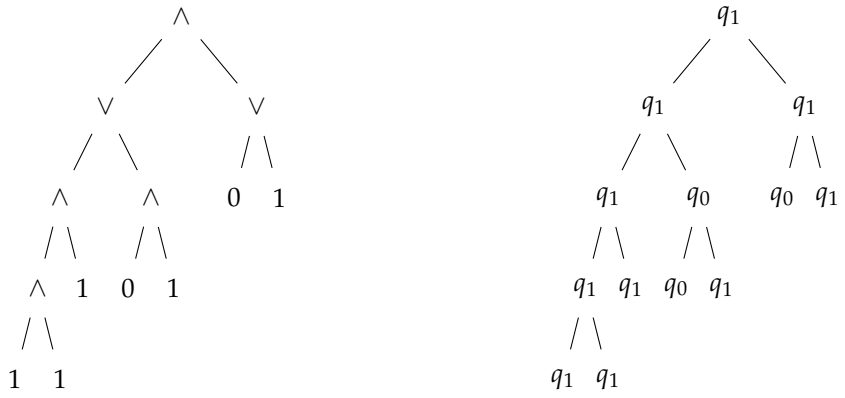
Note that by reversing the transitions of N↓TAs, and regarding the final combinations as initial combinations, one obtains a tree automaton that proceeds in a dual way to the corresponding N↓TA. The tree automaton starts its computation at the leaves of the tree and works its way up to the root, and is thus called a *nondeterministic bottom up tree automaton (N↑TA)*. Both N↓TAs and N↑TAs recognize the same class of tree languages.

Example 2.4. Consider the N↓TA $\mathcal{A} = (Q, M, q_0, \Delta, F)$ over the alphabet $M = \{0, 1, \wedge, \vee\}$ and state set $Q = \{q_0, q_1\}$, initial state q_1, final combinations $F = \{(q_0, 0), (q_1, 1)\}$ and transition relation

$$\begin{aligned}\Delta = \{&(q_1, \vee, q_1, q_1), (q_1, \vee, q_1, q_0), (q_1, \vee, q_0, q_1), (q_0, \vee, q_0, q_0),\\ &(q_1, \wedge, q_1, q_1), (q_0, \wedge, q_1, q_0), (q_0, \wedge, q_0, q_1), (q_0, \wedge, q_0, q_0)\ \}.\end{aligned}$$ ◀

The trees over M correspond to Boolean expressions (without negation), and N↓TA $\mathcal{A}$ accepts the set of trees representing these Boolean expressions that evaluate to true. A tree $\tau \in T_M$ corresponding to the Boolean expression $(((1 \wedge 1) \wedge$

Figure 2.2 A run of the N↑TA $\mathcal{A}$ on τ of Example 2.4.



(a) Tree τ. (b) Accepting run of $\mathcal{A}$ on τ.

$1) \vee (0 \wedge 1)) \wedge (0 \vee 1) = 1$ and the accepting run of $\mathcal{A}$ on τ are depicted in Figure 2.2.

ω-Automata

We now extend the results of the previous paragraph to infinite words and introduce the framework of ω-automata originating back to Büchi [Büc60] and Muller [Mul63]. As for automata over finite words, this involves two aspects: First, a transition system is used to specify elementary steps which are now carried out in a nonterminating process. Secondly, the infinite runs generated this way are checked with an acceptance condition. Usually, these conditions refer to those states which are visited infinitely often within a run. Different uses of these states lead to a broad spectrum of acceptance conditions for infinite words. Towards defining these acceptance conditions, we denote for an ω-word α over an alphabet M the number of occurrences of a letter $a \in M$ in α by $|\alpha|_a$. Given an ω-word $\alpha \in M^\omega$, let

$$Occ(\alpha) = \{a \in M \mid \exists s\, \alpha(s) = a\}$$

be the (possibly finite) set of letters occurring in α, and

$$Inf(\alpha) = \{a \in M \mid \forall s \exists t > s\, \alpha(t) = a\}$$

be the (possibly finite) set of letters occurring infinitely often in α.

As stated above, the usual definitions of deterministic and nondeterministic automata are adapted to the case of ω-input-words by the introduction of new acceptance conditions. For this purpose one introduces an "acceptance component" in the specification of automata, which will arise in different formats.

An *ω-automaton* $\mathcal{A}$ is a quintuple $\mathcal{A} = (Q, M, q_0, \Delta, Acc)$, where Q is a finite set of states, M is an alphabet, $q_0 \in Q$ is the initial state, $\Delta \subseteq Q \times M \times Q$ is the transition relation, and Acc is the acceptance component. In a *deterministic ω-automaton*, a transition function $\delta : Q \times M \to Q$ is used.

The notion of a run ρ of $\mathcal{A}$ on a word $\alpha \in M^\omega$ is defined in analogy to the case of automata over finite words, where a run on infinite words is of course again infinite. The acceptance component can be given as a set of states, as a set of state-sets, or as a function from the set of states to a finite set of natural numbers. We will only introduce some instances, for a more detailed discussion, we refer the reader to [Tho90] and [GTW02].

An ω-automaton $\mathcal{A} = (Q, M, q_0, \Delta, F)$ with acceptance component $F \subseteq Q$ is called a *Büchi automaton* if it is used with the following Büchi acceptance condition: An ω-word $\alpha \in M^\omega$ is accepted by $\mathcal{A}$ iff there exists a run ρ of $\mathcal{A}$ satisfying the condition

$$Inf(\rho) \cap F \neq \emptyset,$$

i.e., at least one of the states in F has to be visited infinitely often during the run.

Büchi recognizable ω-languages are closed under union, complementation, intersection, and projection (as shown by Büchi [Büc62]; a proof can be found in [Tho90]), and the nonemptiness problem for Büchi automata is decidable. Since they also correlate with the ω-languages generated by ω-regular expressions, Büchi recognizable ω-languages are called *regular ω-languages*.

Let us associate indices (called colors) with states of an ω-automaton. An ω-automaton $\mathcal{A} = (Q, M, q_0, \Delta, c)$ with acceptance component $c : Q \to \{0, \ldots, k\}$ with $k \in \mathbb{N}$ is called a *parity automaton* if it is used with the following parity acceptance condition: An ω-word $\alpha \in M^\omega$ is accepted by $\mathcal{A}$ iff there exists a run ρ of $\mathcal{A}$ satisfying the condition

$$\min\{c(q) \mid q \in Inf(\rho)\} \text{ is even.}$$

It is well-known that nondeterministic Büchi automata and deterministic parity automata are equivalent in expressive power, i.e., they recognize the same ω-languages (cf. e.g. [GTW02]).

Let us recall the simple fact that deterministic parity automata are closed under complement by increasing the coloring by 1. We shall also use the reduction of alternating parity ω-automata to deterministic ones (cf. [MS84, MH84]).

The formalism of ω-automata on infinite trees is again a straightforward extension of automata on finite trees and ω-automata on infinite words as introduced above. We will start with Büchi tree automata, which were first studied by Rabin [Rab70].

A *Büchi tree automaton* over the alphabet M is of the form $\mathcal{A} = (Q, M, q_0, \Delta, F)$ with finite state set Q, initial state $q_0 \in Q$, transition relation $\Delta \subseteq Q \times M \times Q \times Q$,

and a set $F \subseteq Q$ of final states. A run ρ of $\mathcal{A}$ on a tree $\tau \in T_M^\omega$ is a map $\rho : \{0,1\}^* \to Q$ with $\rho(\varepsilon) = q_0$ and $(\rho(w), \tau(w), \rho(w0), \rho(w1)) \in \Delta$ for $w \in \{0,1\}^*$. The run ρ is successful of on each path some final state occurs infinitely often.

A *Rabin tree automaton* over the alphabet M has the form $\mathcal{A} = (Q, M, q_0, \Delta, \Omega)$ where Q, q_0, and Δ are as before and $\Omega = \{(E_1, F_1), \ldots, (E_n, F_n)\}$ is a collection of "accepting pairs" of state sets $E_i, F_i \subseteq Q$. A run ρ of the Rabin tree automaton $\mathcal{A}$ is successful if for all paths π and the run on this path $\rho|_\pi$, there exists an $i \in \{1, \ldots, n\}$ such that

$$Inf(\rho|_\pi) \cap E_i = \emptyset \text{ and } Inf(\rho|_\pi) \cap F_i \neq \emptyset.$$

A tree $\tau \in T_M^\omega$ is accepted by the Büchi resp. Rabin tree automaton $\mathcal{A}$ if some run of $\mathcal{A}$ on τ is successful in the respective sense. A set $T \subseteq T_M^\omega$ is Büchi recognizable resp. Rabin recognizable, if it consists of the trees accepted by a Büchi resp. Rabin tree automaton.

Since any Büchi tree automaton may be regarded as a Rabin tree automaton (set $\Omega = \{(\emptyset, F)\}$), any Büchi recognizable set of infinite trees is Rabin recognizable. On the other hand, Rabin tree automata have more expressive power than Büchi tree automata, cf. e.g. [Tho90]. As is well-known, the parity condition can also be applied for tree automata and yields the model of parity tree automata equivalent to Rabin tree automata. For each of these models of tree automata, nonemptiness can be decided, and (by Rabin's "basis theorem" [Rab72]) a nonempty Rabin recognizable tree language contains a regular tree (see e.g. [GTW02]).

The seminal paper [Rab69] that introduced Rabin tree automata provided as its main result the expressive equivalence between Rabin tree automata and monadic second-order logic over labeled binary trees, from which – by the mentioned nonemptiness test – it follows that the monadic second-order theory of the unlabeled binary tree is decidable ("Rabin's tree theorem").

Chapter 3

Path Logic and Automata over Binary Trees

In this chapter we aim at clarifying the expressive power of chain logic as a fragment of MSO-logic over binary trees, in the automata-theoretic framework.

As mentioned in the Introduction, chain logic is an interesting subsystem of MSO-logic, as far as applications in the verification of nonterminating systems are concerned. Properties of these systems are expressed in terms of infinite system-runs, i.e., infinite sequences. So standard formalisms used in verification involve path quantifiers over paths, respectively chains in trees, most prominently in the systems CTL, CTL*, and ECTL* (see e.g. [Tho90]).

Despite the focus on path quantification in dozens and even hundreds of papers, the automata-theoretic treatment of these logics in the literature refers to the general model of Rabin tree automata or parity tree automata that captures the much stronger MSO-logic. In this chapter we develop a model of tree automaton that captures chain logic over binary trees. Its features are summarized as follows: We use

- "full transitions" that prescribe a state *and a letter* for triples of nodes (parent, left son, right son),
- an alternating mode of operation with existential and universal states,
- a rank function for the states with values in $\mathbb{N}$ which allows to keep a rank only in one direction, while decreasing it in the other.

Since for these *ranked alternating tree automata* there is (up to our knowledge) no easy way of obtaining nondeterministic automata that capture chain logic, the applicability of our model in problems on the expressive power of chain logic seems to be limited.

The chapter is divided into three sections: In the first we recall path logic and chain logic (including its version "path-guarded chain logic"), in the second we develop the model of ranked alternating tree automata, and in the third we show how to transform formulas of path-guarded chain logic into such automata and that tree languages recognized by ranked alternating tree automata are in fact definable in chain logic.

3.1 Path Logic and Chain Logic

In this section we consider binary labeled trees where the labels are from a set $\{0,1\}^n$. We capture such a labeling by a tuple $\overline{P}$ of predicates $(P_1, \ldots, P_n)$ and write $(\tau, \overline{P})$; here τ stands for the structure

$$\tau = (\{0,1\}^*, \varepsilon, \cdot 0, \cdot 1).$$

As mentioned earlier, a path is a subset of $\{0,1\}^*$ that is linearly ordered by the prefix relation $\preceq$ and also maximal with this property. A chain is a subset of a path. We associate to each chain C an "induced path" P^C. For an infinite chain C this is the unique path P^C with $C \subseteq P^C$. If C is finite with last element w then P^C contains all prefixes of $w0^\omega$. If $C = \emptyset$ then $P^C = 0^\omega$.

Let us introduce path logic and chain logic in a format that is useful for the subsequent investigations.

We begin with path logic over models $(\tau, \overline{P})$. Path logic is MSO-logic where second-order quantifications range over paths.

Chain logic over models $(\tau, \overline{P})$ is MSO-logic where set quantifications range over chains.

In order to simplify some of the treatments below, we consider MSO-logic, now adapted to chain logic, in a version where first-order variables and quantifiers are eliminated. Over models $(\tau, \overline{P})$, this formalism has the atomic formulas

$$S \subseteq T \quad S \subseteq P \quad \mathrm{Sing}(S) \quad Suc_0(S,T) \quad Suc_1(S,T) \quad S < T$$

with the standard meaning of $\subseteq$ and the following conventions

- $\mathrm{Sing}(S)$: S is a singleton,
- $Suc_i(S,T)$: S, T are singletons $\{s\}$, $\{t\}$ with $Suc_i(s,t)$ for $i = 0, 1$,
- $S < T$: S, T are singletons $\{s\}$, $\{t\}$ with $s < t$.

Remark 3.1. As noted for MSO-logic over word models, the partial order relation $<$ over trees is definable in terms of the successor relations Suc_i. Additionally, the singleton property Sing can be expressed in terms of the subset relation $\subseteq$, using the equivalence $\mathrm{Sing}(S) \leftrightarrow$ "there is precisely one proper subset":

$$\begin{aligned}\mathrm{Sing}(S) \equiv \exists T(T \subseteq S \wedge \neg T = S) \wedge \\ \neg\exists T_1 \exists T_2(\neg T_1 = T_2 \wedge \neg T_1 = S \wedge \neg T_2 = S \wedge T_1 \subseteq S \wedge T_2 \subseteq S)\end{aligned}$$

with $S = T \leftrightarrow S \subseteq T \wedge T \subseteq S$. ◂

The next step in fixing an appropriate syntax is to introduce a "guarded" version of chain logic. Here we use the fact that each chain arises as a subset of a suitable path. If chain C is finite, we may fix this path as the leftmost one containing C. This also applies to singleton sets. We introduce *path-guarded chain logic* in which each chain quantifier $\exists S^C$, $\forall S^C$ is introduced as a subset of a path, i.e., in the format $\exists T^P \exists S^C \subseteq T^P$, respectively $\forall T^P \forall S^C \subseteq T^P$. Formally, this notation serves as a short notation for

$$\exists T^P\big(\exists S^C(S^C \subseteq T^P \wedge \ldots)\big), \forall T^P\big(\forall S^C(S^C \subseteq T^P \rightarrow \ldots)\big), \text{respectively.}$$

Since each path is a (special case of a) chain, each formula of path-guarded chain logic can be written as a chain logic formula; for this one only has to note that the property of being a path can be defined in chain logic by

$$\varphi(S) = \neg\exists T(S \subseteq T \wedge \neg S = T)$$

meaning that there is no proper superset that again is a chain.

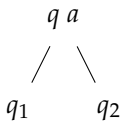
Proposition 3.2. *Chain logic is expressively equivalent to path-guarded chain logic, i.e., for each chain logic formula, we can construct an equivalent path-guarded chain logic formula and vice versa.*

It is convenient to indicate by notation whether for a given variable S the intended interpretation is by chains or by paths. We do this by writing S^C, resp. S^P.

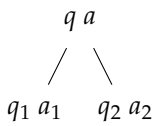
3.2 Ranked Alternating Tree Automata

In this section we introduce a model of automaton over trees that captures the expressive power of chain logic. We restrict here to the case of binary trees. Mostly we focus on the model of the binary *infinite* tree. But by a straightforward adaption of the arguments we obtain our results for finite binary trees.

In our automaton model, we work with transitions that are a slight variant of the standard transitions as used in the automata introduced by Rabin [Rab69]. Where Rabin uses transitions of the form



indicating that in state q on a vertex labeled a the automaton may proceed to the left into state q_1 and to the right into state q_2, we work with *full transitions*



which can be applied according to the presence of symbol a_1, resp. a_2, at the two successors of the current vertex. We call these transitions *full*; they involve more overlapping in runs of tree automata, in close analogy to the "tiling systems" (cf. Thomas [Tho91], Giammarresi et al. [GRST96]) over two dimensional pictures. For the model of Rabin automaton, the shift to full transitions is inessential. However, this shift turns out relevant in the treatment of path logic and chain logic.

A second special feature introduced in the sequel is the introduction of ranks. To each state we associate a natural number as a rank, and the idea in applying this concept is to require in a successful run ρ that

- on each path the ranks decrease

  (i.e., for tree nodes $v \leq w$ we have $rank(\rho(v)) \geq rank(\rho(w))$),

- in each transition, at most one branch (left or right) allows to keep the rank, while on the other a strict decrement applies – excepting of course the case of rank 0,

- finally, we refer to alternating automata, rather than nondeterministic automata as in the classical work of Rabin [Rab69].

As a preparation of a formal definition, let us fix the concept of nondeterministic tree automata with full transitions:

Definition 3.3. A tree automaton $\mathcal{A}$ with full transitions is of the form $\mathcal{A} = (Q, M, q_0, \Delta, \Omega)$ where

- Q is the set of final states,
- M is the input alphabet,
- $q_0 \in Q$ is the initial state,
- $\Delta \subseteq (Q \times M)^3$ is the transition relation, and
- Ω is the acceptance component (to be specified separately; if Ω is not specified explicitly, we assume the parity condition).

A *run* of $\mathcal{A}$ on a tree $(\tau, \overline{P})$ is a map

$$\rho : \{0,1\}^* \to Q$$

such that for each vertex $v \in \{0,1\}^*$

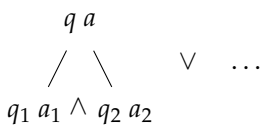
$$\big((\rho(v),\tau(v)),(\rho(v0),\tau(v0)),(\rho(v1),\tau(v1))\big) \in \Delta.$$

The automaton is *deterministic*, if for each $q \in Q, a, a_1, a_2 \in M$ there is precisely one transition $\big((q,a),(q_1,a_1),(q_2,a_2)\big) \in \Delta$.

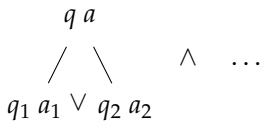
In *alternating tree automata*, the set Q of states contains existential states (collected in a subset $Q_\exists$ of Q) and universal states (collected in a subset $Q_\forall$ of Q), as well as states of rank 0 and states for ε-transitions (collected in $Q^\varepsilon_\exists$ resp. $Q^\varepsilon_\forall$). The transitions for a state in $Q_\exists \cup Q_\forall$ are given as a list $\{\vartheta_1, \ldots, \vartheta_k\}$ where $\vartheta_i \in (Q \times M)^3$. In a run, at a state $q \in Q_\exists$, one transition of the list needs to be chosen (thus, we may think of the list as a disjunction), while at a state $q \in Q_\forall$, all transitions of the list need to be applied (thus leading to a conjunction of the given transitions). Semantically, transitions for a state $\in Q_\exists$ can be viewed as containing an implicit conjunction, as is the case in the usual semantics of tree automata transitions (i.e., both directions are pursued). In order to obtain a dual transition to this, we define for a state in $Q_\forall$, that it implicitly contains a *disjunction* over the two successors, i.e., here only one direction is pursued.

Intuitively, we may indicate the use of the transitions as follows:

- for $q \in Q_\exists$:



- for $q \in Q_\forall$:



For states of rank 0, collected in a subset Q_0 of Q, we require the states to generate a deterministic automaton in the sense described above. For states for ε-transitions, we again provide a list of transitions $\subseteq 2^{(Q\times M\times Q)}$ that will either be connected disjunctively or conjunctively. All lists of transitions may consist of one element only. Formally, a disjunction or conjunction may be restored using the state REJECT, resp. ACCEPT introduced below for the second disjunction, resp. conjunction member in order to force usage of the first.

In our model of ranked alternating tree automata, for technical reasons, we connect the use of ε-transitions with corresponding ε-ranks of states: we intro-

duce subsets $Q^{\varepsilon}_{\exists}, Q^{\varepsilon}_{\forall} \subseteq Q$ for the activation of existential, resp. universal ε-transitions. For each application of an ε-transition, the ε-rank has to decrease for all target states. Thus, infinite chains of ε-transitions are excluded.

Finally, we introduce the notion of ranked alternating tree automata (with full transitions):

Definition 3.4. A ranked alternating tree automaton with full transitions is of the form $\mathcal{A} = (Q, M, q_0, \Delta, \Omega, \mathit{rank}, \varepsilon\text{-}\mathit{rank})$ where

- M, q_0, Ω are as above,
- $Q = Q_{\exists} \uplus Q_{\forall} \uplus Q_0 \uplus Q^{\varepsilon}_{\exists} \uplus Q^{\varepsilon}_{\forall}$,
- $\Delta \subseteq 2^{(Q\times M)^3} \cup 2^{(Q\times M\times Q)}$, where for states in $Q_{\exists} \cup Q_{\forall} \cup Q_0$, we have finite transition lists in $2^{(Q\times M)^3}$, while for states in $Q^{\varepsilon}_{\exists} \cup Q^{\varepsilon}_{\forall}$, we have finite transition lists in $2^{(Q\times M\times Q)}$. For states in $Q_{\exists} \cup Q^{\varepsilon}_{\exists}$, these lists are understood as disjunctions, for states in $Q_{\forall} \cup Q^{\varepsilon}_{\forall}$ as conjunctions, and for states in Q_0, we allow precisely one transition in $(Q \times M)^3$.

  Again, note that the semantics for transitions of existential and universal state differ: As we have the usual "implicit conjunction" over the two successors for a transition for existential states, we work with an implicit disjunction over the two successors for a transition for universal states, i.e., *one* direction needs to be followed.

- $\mathit{rank} : Q \to \mathbb{N}$ is the rank function, and for each occurring transition

$$\begin{array}{ccc} & q\ a & \\ \swarrow & & \searrow \\ q_1\ a_1 & & q_2\ a_2 \end{array}$$

  we have that $\mathit{rank}(q) \geq \mathit{rank}(q_1), \mathit{rank}(q_2)$, and equality holds at most in one case (unless $\mathit{rank}(q) = 0$ in which case $\mathit{rank}(q_1) = \mathit{rank}(q_2) = 0$). In each case, the switch from a state in $Q_{\exists}$ to a state in $Q_{\forall}$ or vice versa is only possible by a proper decrease of the rank.

- For technical reasons, we introduce the function $\varepsilon\text{-}\mathit{rank} : Q \to \mathbb{N}$, such that in each ε-transition

$$q\ a \to q'$$

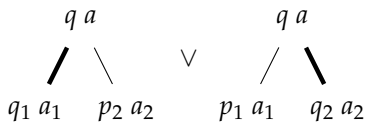
  it holds that $\varepsilon\text{-}\mathit{rank}(q) > \varepsilon\text{-}\mathit{rank}(q')$. This restriction guarantees that there are no infinite ε-chains within an automaton run (since ranks never increase and the minimum rank and ε-rank is 0).

For rank 0, we allow only the states ACCEPT and REJECT which both are reproduced without change in transitions, regardless of which input letters occur. In the context of the parity condition, ACCEPT carries color 0 and REJECT carries color 1. We also denote by ACCEPT, resp. REJECT the corresponding one-state tree automata – so $T(\text{ACCEPT}) = T_M^\omega$ and $T(\text{REJECT}) = \emptyset$.

As the rank-sum of an automaton we take the sum of the rank and ε-rank of its initial state. A tree $(\tau, \overline{P})$ is accepted by $\mathcal{A}$ if there exists a run tree as defined below of $\mathcal{A}$ on $(\tau, \overline{P})$ such that the state sequence along each path satisfies the acceptance condition. We denote by $T(\mathcal{A})$ the set of trees accepted by $\mathcal{A}$.

In order to make transitions in $(Q \times M)^3$ more "readable", we indicate the direction in which the rank is preserved by a boldface branch (and we use no boldface line if the rank decreases in both directions).

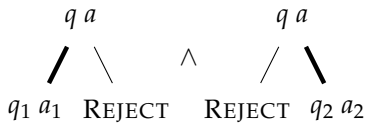
For example, for $q \in Q_\exists$ with $rank(q) > 0$ and a list of two transitions, the transitions may have the form



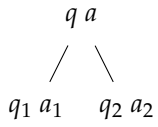
where we have $rank(q) \geq rank(q_1), rank(q_2)$, and, moreover, $rank(q) > rank(p_2)$ and $rank(q) > rank(p_1)$.

The two cases $rank(q) = rank(q_1)$, $rank(q) = rank(q_2)$ correspond to the fact that a path under consideration is pursued to the left or to the right. Similarly for universal states, paths through both successor nodes are pursued.

We note that the standard model of *deterministic tree automaton* (considered here, however, with full transitions) can be captured by a universal ranked tree automaton with transitions



as an implementation of transitions



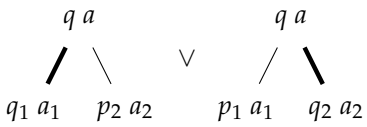
Let us turn to the definition of acceptance, following the pattern that was introduced in the work of Muller and Schupp [MS84].

A *run tree* σ of $\mathcal{A}$ over the input tree $(\tau, \overline{P})$ is built up from nodes

$$(v, q, a)$$

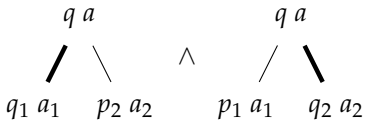
indicating that the automaton is at node v in state q, reading a at v.

The root of σ is labeled (ε, q_0, a) for the symbol $a = \tau(\varepsilon)$. To increase readability, we indicate the structure of the run tree for two disjunction resp. conjunction members, the treatment of more members is straightforward. Given a vertex (v, q, a) with $q \in Q_\exists$, and $\tau(v0) = a_1$, $\tau(v1) = a_2$, the two successors in σ are labeled $(v0, q_1, a_1)$, $(v1, p_2, a_2)$ or $(v0, p_1, a_1)$, $(v1, q_2, a_2)$ if the transition



exists in $\mathcal{A}$. This accounts for the semantics that for a list of transitions of an existential state, we choose one transition with an implicit conjunction, thus following both directions of the transition.

Given a vertex (v, q, a) with $q \in Q_\forall$, and $\tau(v0) = a_1, \tau(v1) = a_2$, we need to reflect the semantics in a dual way, as already mentioned above: for each transition in the list, we follow one direction, i.e., in σ, we have here either $(v0, q_1, a_1)$ and $(v0, p_1, a_1)$ as successors of (v, q, a), or $(v0, q_1, a_1)$ and $(v1, q_2, a_2)$, or $(v1, p_2, a_2)$ and $(v0, p_1, a_1)$, or $(v1, p_2, a_2)$ and $(v1, q_2, a_2)$ if the transition



exists in $\mathcal{A}$. The reader may silently insert a disjunction between the two brother nodes in the two transitions in order to remind himself of the semantics of universal transitions.

If for a vertex (v, q, a) the state q is in $Q^\varepsilon_\exists$, we obtain one successor node (v, q', a) for a transition $q\,a \rightarrow q'$of the list, and for a state q in $Q^\varepsilon_\forall$, we get as many successors (v, q', a) as there are entries in the list of transitions of q.

Remark 3.5. For a ranked alternating tree automaton $\mathcal{A}$ with the parity acceptance condition, the dual automaton $\widetilde{\mathcal{A}}$ is obtained by

- exchanging $Q_\exists$ and $Q_\forall$, resp. $Q^\varepsilon_\exists$ and $Q^\varepsilon_\forall$
- exchanging ACCEPT and REJECT,
- changing the coloring c to $c + 1$ (where $c + 1(q) := c(q) + 1$).

Note that the dualization does not affect the ranks. ◂

With the result of Muller and Schupp [MS87], we can state the following result over dualization of ranked alternating tree automata: From the definitions it is immediate that we have the following duality result:

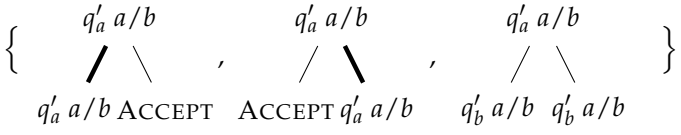
Theorem 3.6. *Given a ranked alternating parity tree automaton $\mathcal{A}$, the* dual *automaton $\widetilde{\mathcal{A}}$ recognizes the complement of the tree language recognized by $\mathcal{A}$. The same result holds for ranked alternating tree automata over finite binary trees and complementation with respect to the set of all finite binary trees.*

Example 3.7. Consider the tree language $T_0 \subseteq T^{\omega}_{\{a,b\}}$ with

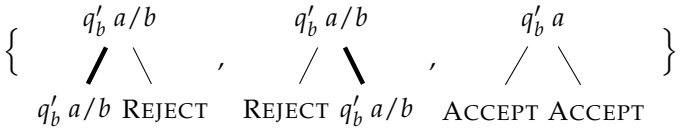
$$\tau \in T_0 \Leftrightarrow \text{for all vertices } v \text{ labeled } a \text{ there is a path } \pi \text{ starting in } v \text{ fully labeled } b \text{ for the proper descendants of } v \text{ on } \pi\}$$

As a more intuitive approach, we consider the complement $\overline{T_0}$ of T_0 first: we then need to search nondeterministically for a node u labeled a such that all the paths starting in u do not only consist of bs. This can be accomplished by $\mathcal{A}' = (\{q'_a, q'_b, \textsc{Accept}, \textsc{Reject}\}, \{a,b\}, q'_a, \Delta', c', \mathit{rank}', \varepsilon\text{-}\mathit{rank}')$ where $q'_a \in Q'_{\exists}$ nondeterministically searches for the a-node and $q'_b \in Q'_{\forall}$ checks all paths starting there. The details of the automaton are:

- for the existential state q'_a we have the list of transitions

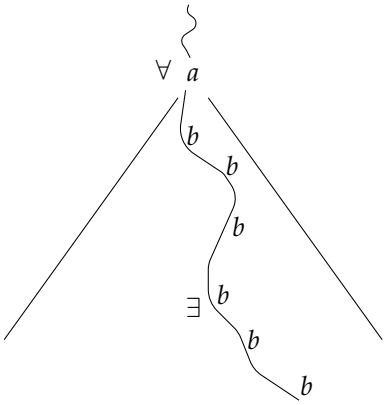


- for the universal state q'_b we have the list of transitions



- for the acceptance condition, we let $c'(q'_a) = c'(q'_b) = 1$ (where – as defined above – $c'(\textsc{Accept}) = 0$ and $c'(\textsc{Reject}) = 1$),
- we set $\mathit{rank}'(q'_a) = 2$ and $\mathit{rank}'(q'_b) = 1$ (again, as defined above we have $\mathit{rank}'(\textsc{Accept}) = \mathit{rank}'(\textsc{Reject}) = 0$),
- and finally, since we do not employ ε-transitions, we have $\varepsilon\text{-}\mathit{rank}'(q'_a) = \varepsilon\text{-}\mathit{rank}'(q'_b) = \varepsilon\text{-}\mathit{rank}'(\textsc{Accept}) = \varepsilon\text{-}\mathit{rank}'(\textsc{Reject}) = 0$.

Figure 3.1 Indication of a run on a tree $\tau \in T_0$ from Example 3.7.



It is not hard to see that $T(\mathcal{A}') = \overline{T_0}$. Now, constructing the dual ranked alternating tree automaton $\widetilde{\mathcal{A}'}$ to $\mathcal{A}'$ yields an automaton recognizing $\overline{T_0}$.

On the other hand, we can also construct a ranked alternating tree automaton $\mathcal{A}$ recognizing T_0 directly. It has to check the situation illustrated as depicted in Figure 3.1.

We use universal branching from initial state q_a to detect all nodes carrying letter a. Meeting letter a we universally add a branch to check for a path containing letters b only (via q_b). Thus, we construct the ranked alternating tree automaton $\mathcal{A} = (\{q_a, q_b, \text{ACCEPT}, \text{REJECT}\}, \{a, b\}, q_a, \Delta, c, \mathit{rank}, \varepsilon\text{-}\mathit{rank})$ with the following details:

- $q_a \in Q_\forall$ and $q_b \in Q_\exists$,
- for the universal state q_a we have the list of transitions

$$\Big\{ \begin{array}{ccccc} q_a\ a/b & & q_a\ a/b & & q_a\ a \\ / \quad \backslash & , & / \quad \backslash & , & / \quad \backslash \\ q_a\ a/b\ \text{REJECT} & & \text{REJECT}\ q_a\ a/b & & q_b\ a/b\ \ q_b\ a/b \end{array} \Big\}$$

- for the existential state q_b we have the list of transitions

$$\Big\{ \begin{array}{ccc} q_b\ b & & q_b\ b \\ / \quad \backslash & , & / \quad \backslash \\ q_b\ b\ \text{ACCEPT} & & \text{ACCEPT}\ q_b\ b \end{array} \Big\}$$

- for the acceptance condition, we let $c(q_a) = c(q_b) = 0$ (where as defined above $c(\text{ACCEPT}) = 0$ and $c(\text{REJECT}) = 1$),

- we set $rank(q_a) = 2$ and $rank(q_b) = 1$ (and as defined above $rank(\textsc{Accept}) = rank(\textsc{Reject}) = 0$),
- and finally, $\varepsilon\text{-}rank(q_a) = \varepsilon\text{-}rank(q_b) = \varepsilon\text{-}rank(\textsc{Accept}) = \varepsilon\text{-}rank(\textsc{Reject}) = 0$.

It is not hard to see that $\widetilde{\mathcal{A}'}$ and $\mathcal{A}$ both recognize T_0. ◂

3.3 From Chain Logic to Automata and Back

The purpose of this section is to establish a transformation of formulas of path-guarded chain logic to ranked alternating automata. As remarked before, the restriction to path-guarded formulas is inessential. More formally, we show:

Theorem 3.8. *For any formula of path-guarded chain logic $\varphi(T_1^P, S_1^C, \ldots, T_k^P, S_k^C)$ with variables $T_1^P, \ldots, T_k^P$ for paths and $S_1^C, \ldots, S_k^C$ for chains such that $S_i^C \subseteq T_i^P$ for $1 \leq i \leq k$ and unary predicate symbols $\overline{Z} = (Z_1, \ldots, Z_m)$, one can construct a ranked alternating tree automaton $\mathcal{A}_\varphi$ over the alphabet $\Sigma_{k,m} = \{0,1\}^{2k} \times \{0,1\}^m$ such that for each k-tuple $(M_1^P, \ldots, M_k^P)$ of paths, each k-tuple $(M_1^C, \ldots, M_k^C)$ of chains with $M_i^C \subseteq M_i^P$ for $1 \leq i \leq k$ and each tuple $\overline{K} = (K_1, \ldots, K_m)$ of subsets of the tree domain $\{0,1\}^*$ we have*

$$(\tau, \overline{M^P}, \overline{M^C}, \overline{K}) \models \varphi \text{ iff } \mathcal{A}_\varphi \text{ accepts } (\tau, \overline{M^P}, \overline{M^C}, \overline{K}).$$

The claim holds with the obvious modifications also over finite trees.

As a preparation for the proof, let us fix some notations first: For a state $q_i \in Q_\exists \cup Q_\forall$, we denote the list of transitions available to q_i as

$$\Big\{ \begin{array}{ccc} q_i\ \overline{b}_1 & & q_i\ \overline{b}_2 \\ \diagup\ \diagdown & , & \diagup\ \diagdown \\ q^i_{11}\ \overline{b}_{11}\ \ q^i_{12}\ \overline{b}_{12} & & q^i_{21}\ \overline{c}_{21}\ \ q^i_{22}\ \overline{c}_{22} \end{array} \quad , \ \cdots\ , \quad \begin{array}{c} q_i\ \overline{b}_{k_i} \\ \diagup\ \diagdown \\ q^i_{k_i 1}\ \overline{c}_{k_i 1}\ q^i_{k_i 2}\ \overline{c}_{k_i 2} \end{array} \Big\},$$

and for a state $q_i \in Q^\varepsilon_\exists \cup Q^\varepsilon_\forall$, we denote the list of ε-transitions available to q_i as

$$\{q_i\ \overline{b}_1 \to q^i_1, q_i\ \overline{b}_2 \to q^i_2, \ldots, q_i\ \overline{b}_{k_i} \to q^i_{k_i}\}.$$

Since for a quantified path, we want to refer to the first two bits (for the path component T^P and the chain component S^C) of the letters at each state, we will denote those as b_ℓ and b'_ℓ for letter vector $\overline{b}_\ell$, respectively $c_{\ell m}$ and $c'_{\ell m}$ for letter vector $\overline{c}_{\ell m}$ (for $1 \leq \ell \leq k_i$ and $m \in \{1,2\}$).

For the proof we employ an induction over the structure of path guarded chain logic formulas. Thus, the proof of Theorem 3.8 then proceeds in three stages:

- atomic formulas (inclusion and successor),
- Boolean connectives (negation and disjunction), and
- quantification (existential quantification);

  applied to quantifier pairs ($\exists T^P \exists S^C \subseteq T^P$).

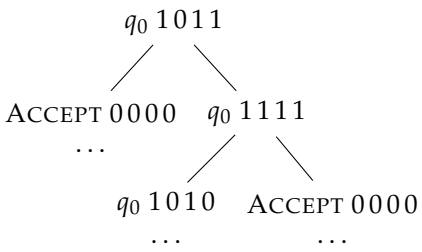
Note that these three cases suffice since all other connectives and quantifiers are expressible in terms of them. While the first two cases are rather straightforward constructions, existential quantification demands a nontrivial step.

Lemma 3.9. *Theorem 3.8 holds for atomic formulas.*

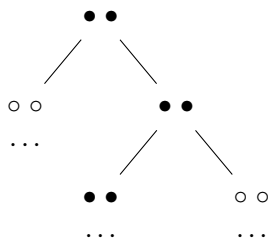
PROOF. It is convenient to rewrite atomic formulas as stated in Remark 3.1. For each of the remaining cases (a) inclusion and (b) successor relation, we consider the chain variables in connection with their guarding path variable.

(a) Inclusion: $S_1^C \subseteq S_2^C \wedge S_1^C \subseteq T_1^P \wedge S_2^C \subseteq T_2^P$ $(*)$

We present the desired tree automaton discarding the alphabet components for other variables than T_1^P, T_2^P, S_1^C, S_2^C. The alphabet is thus $\{0,1\}^4$, and we list the components in the order T_1^P, S_1^C, T_2^P, S_2^C. We use a ranked alternating tree automaton that has transitions capturing all labellings that are admissible under $(*)$. First, we indicate a typical run:



with the following two chains indicated by the full bullets:



So we require that the label $(1,*,*,*)$ induces $(1,*,*,*)$ at precisely one of the two sons, similarly for $(*,*,1,*)$, and also for $(1,b_1,1,b_2)$ we stipulate $b_1 \leq b_2$ (i.e., $b_1 = 1$, $b_2 = 0$ is excluded) which can all be checked via one accepting state $q_0 \in Q_\exists$. If we encounter a label $(1,0,1,1)$, we switch to rejecting state REJECT and on all other paths, we preserve the label $(0,0,0,0)$ via ACCEPT.

Note that if one or both chains are finite, the situation changes due to our convention that the subsuming path is chosen to be the leftmost path. To ease readability, we omit the treatment of these cases as they can be captured with some technical effort.

(b) For the atomic formulas $Suc_0(S_1,S_2)$ and $Suc_1(S_1,S_2)$, the construction is similar; clearly a deterministic automaton can check this condition. ■

Lemma 3.10. *Given path guarded chain logic formulas φ_1 and φ_2 and their equivalent ranked alternating tree automata $\mathcal{A}_{\varphi_1}$ and $\mathcal{A}_{\varphi_2}$, we can construct ranked alternating tree automata equivalent to $\neg\varphi_1$ and $\varphi_1 \vee \varphi_2$.*

PROOF. For a given alternating tree automaton $\mathcal{A}_{\varphi_1}$, we can construct the dual automaton $\widetilde{\mathcal{A}}_{\varphi_1}$ as stated in Remark 3.5 which is equivalent to $\neg\varphi_1$.

As for the conjunction $\varphi_1 \vee \varphi_2$, we can exploit the useful formalism of ε-transitions. Therewith, given the two ranked alternating tree automata $\mathcal{A}_{\varphi_1} = (Q^1, M, q_0^1, \Delta^1, \Omega^1, rank^1, \varepsilon\text{-}rank^1)$ and $\mathcal{A}_{\varphi_2} = (Q^2, M, q_0^2, \Delta^2, \Omega^2, rank^2, \varepsilon\text{-}rank^2)$, we can construct a ranked alternating tree automaton $\mathcal{A}_{\varphi_1\vee\varphi_2}$ by introducing a new initial state $q_0 \in Q_\exists^\varepsilon$ which we provide with the transition list

$$\{q_0\ a \to q_0^1, q_0\ a \to q_0^2 \mid a \in M\},$$

where we set $rank(q_0) = \max\{rank(q_0^1), rank(q_0^2)\}$ and accordingly $\varepsilon\text{-}rank(q_0) = \max\{\varepsilon\text{-}rank(q_0^1), \varepsilon\text{-}rank(q_0^2)\} + 1$ (as for the acceptance condition, a state that is traversed once does not modify acceptance over infinite sequences; we may thus e.g. simply color q_0 with 0). Then, $\mathcal{A}_{\varphi_1\vee\varphi_2}$ proceeds as $\mathcal{A}_{\varphi_1}$ resp. $\mathcal{A}_{\varphi_2}$ and thus recognizes the disjunction. ■

Lemma 3.11. *Given a path guarded chain logic formula $\varphi(T^P, S^C)$ and its equivalent alternating tree automaton $\mathcal{A}_\varphi$, one can construct a ranked alternating tree automaton $\mathcal{B}$ equivalent to the path guarded chain logic formula $\exists T^P \exists S^C \subseteq T^P(\varphi(T^P, S^C))$.*

We will proceed in two steps in order to increase readability of the construction. Prior to that, let us describe the idea: With a given ranked alternating tree automaton $\mathcal{A}_\varphi$ of rank-sum $n+m$, the constructed automaton will first guess the quantified path and chain (finite or infinite) using states of rank-sum $n+m+2$,

for which we employ a variation of a powerset construction to conquer universal and existential branching along this path. Thus, we will construct an intermediate ranked alternating tree automaton with sets $P \subseteq Q$ as states on the quantified path. Since our powerset construction will demand the application of an ω-word-automaton along the newly quantified path to check the acceptance condition, a state of this automaton will be added to the sets P in a second step. Thus, pairs (P, p) will be contained in the set of states of the constructed automaton $\mathcal{B}$, all of which will be existential. Apart from one auxiliary construct A^P, states of $\mathcal{A}_\varphi$ will suffice for the rest of the nodes which are not on the quantified path.

Let us define one more auxiliary construction before we continue with the proof of Lemma 3.11. We need the notion of a *final ε-expansion* of a set P of states in order to ease the handling of ε-transitions in our construction.

Definition 3.12. For a given ranked alternating tree automaton $\mathcal{A}$, we define the function $\delta_\varepsilon : Q^\varepsilon_\exists \cup Q^\varepsilon_\forall \times \mathbb{B}^2 \to 2^Q$ as

$$\delta_\varepsilon(q_i, b\,b') := \{q^i_j \mid \text{there exists a transition } q_i\ b\,b' \to q^i_j\}.$$

With δ_ε, we define an *ε-successor* $\widetilde{P}$ of P in a tree τ at node u with $\tau(u) = b\,b'$ as follows:

$$\begin{aligned}\widetilde{P} \text{ is } \varepsilon\text{-successor of } P \Leftrightarrow & \forall q \in Q_\exists \cup Q_\forall : q \in P \Leftrightarrow q \in \widetilde{P} \\ & \delta_\varepsilon(q, b\,b') \subseteq \widetilde{P} \text{ for all } q \in Q^\varepsilon_\forall \cap P \\ & \exists q' \in \widetilde{P} : q' \in \delta_\varepsilon(q, b\,b') \text{ for all } q' \in Q^\varepsilon_\exists \cap P\end{aligned}$$

Now, taking the transitive closure of "being an ε-successor of P" yields the notion of an *ε-descendant* of P which leads us to the definition of the *ε-expansion* $\widetilde{\mathcal{P}}$ of a set P of states at node u in tree τ with $\tau(u) = b\,b'$:

$$\widetilde{\mathcal{P}}(P) = \{P' \mid P' \text{ is an } \varepsilon\text{-descendant of } P\}.$$

Since we are targeting the end of a chain of ε-transitions in our construction, we define the notion of a *final ε-expansion* $\mathcal{P}(P)$ as

$$\mathcal{P}(P) = \{P' \subseteq 2^{Q_\exists \cup Q_\forall} \mid P' \text{ is an } \varepsilon\text{-descendant of } P\}.$$

Now, we turn to the proof of Lemma 3.11.

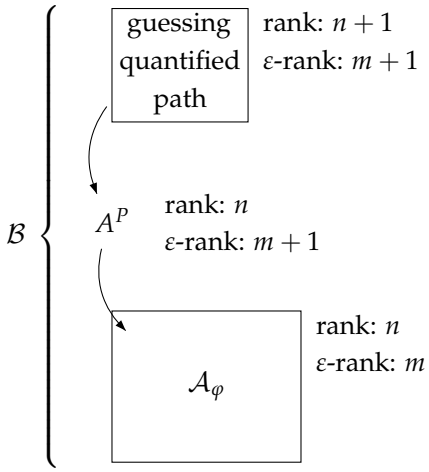
PROOF. Given a ranked alternating tree automaton

$$\mathcal{A}_\varphi = (Q^\mathcal{A}, M \times \{0,1\}^2, q_0^\mathcal{A}, \Delta^\mathcal{A}, \Omega^\mathcal{A}, \mathit{rank}^\mathcal{A}, \varepsilon\text{-}\mathit{rank}^\mathcal{A})$$

with $\mathit{rank}(\mathcal{A}_\varphi) = n$ and $\varepsilon\text{-}\mathit{rank}(\mathcal{A}_\varphi) = m$ and $T(\mathcal{A}_\varphi) = T(\varphi(T^P, S^C))$, the idea of constructing the intermediate ranked alternating tree automaton

$$\mathcal{B} = (Q^\mathcal{B}, M, q_0^\mathcal{B}, \Delta^\mathcal{B}, \Omega^\mathcal{B}, \mathit{rank}^\mathcal{B}, \varepsilon\text{-}\mathit{rank}^\mathcal{B})$$

can be depicted as follows:



Before we consider the role of the ω-word-automaton, we focus on the set components in the states of $\mathcal{B}$. Thus, we start with $\{q_0^{\mathcal{A}}\} := q_0^{\mathcal{B}}$ of rank $n+1$ and ε-rank $m+1$. For a set $P = \{q_1, \ldots, q_k\}$, we build up transitions and successor state sets in two steps:

1. we define how we build up a transition

$$\begin{array}{ccc} & P\,1\,b' & \\ \diagup & & \diagdown \\ P'\,1\,c' & & A^{P'}\,0\,0 \end{array}$$

for $b', c' \in \mathbb{B}$ (thus, the quantified path goes to the left, the other case is analogous):
P' is in the final ε-expansion of $\widehat{P}$ while $\widehat{P}$ and $A^{P'}$ have to fulfill the following conditions:

$$\begin{array}{c} \text{“for all } q_i \in Q_\exists \cap P \text{ exists one transition } (q_i\,1\,b', q^i_{j1}\,1\,c', q^i_{j2}\,0\,0) \\ \text{such that } q^i_{j1} \in \widehat{P} \text{ and } q^i_{j2} \in A^{P'} \\ \text{and} \\ \text{for all } q_i \in Q_\forall \cap P \text{ and for all transitions } (q_i\,1\,b', q^i_{j1}\,1\,c', q^i_{j2}\,0\,0) \\ \text{we have } q^i_{j1} \in \widehat{P} \text{ or } q^i_{j2} \in A^{P'}\text{”} \end{array}$$

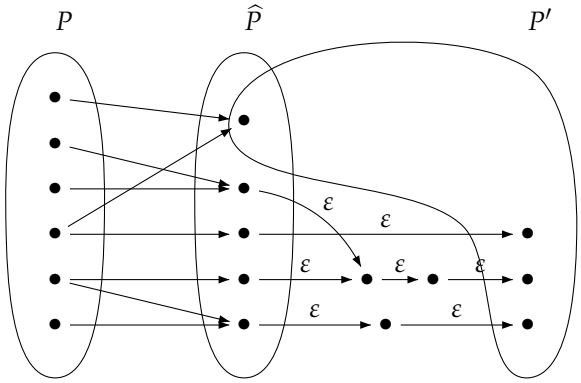
where the ranks are $rank^{\mathcal{B}}(P) = rank^{\mathcal{B}}(P') = n+1$, $rank^{\mathcal{B}}(A^{P'}) = n$, and the ε-ranks are $\varepsilon\text{-}rank^{\mathcal{B}}(P) = \varepsilon\text{-}rank^{\mathcal{B}}(P') = m+1$, $\varepsilon\text{-}rank^{\mathcal{B}}(A^{P'}) = m+1$.

For $A^{P'} = \{p_1, \ldots, p_\ell\}$, we let $A^{P'}$ be in $Q^\varepsilon_\forall$, and define the transition list as $\{A^{P'}\ 0\,0 \to p_1, \ldots, A^{P'}\ 0\,0 \to p_\ell\}$, where $rank^{\mathcal{B}}(p_i) = rank^{\mathcal{A}}(p_i)$, and $\varepsilon\text{-}rank^{\mathcal{B}}(p_i) = \varepsilon\text{-}rank^{\mathcal{A}}(p_i)$ (as before in $\mathcal{A}$).

Therewith, the intermediate ranked alternating tree automaton $\mathcal{B}$ has as "set components" of its states the sets $P, P', A^{P'}$ as defined above, and apart from the quantified path and the intermediate A^P-states, it works as the given automaton $\mathcal{A}$ limited to $\mathcal{A}_{00}$; i.e., since $\mathcal{A}$ had two more input components (of the quantified path), we delete these input components in $\mathcal{A}$ while only keeping those transitions where these input components are 00 (since we only use $\mathcal{A}$ outside of the quantified path).

2. Now, as a second step, we need to take care of the acceptance along the quantified path (for all other paths: "new" parts in $\mathcal{B}$ are at most a finite prefix, thus do not influence acceptance along infinite paths).

   In the sequence of sets P, we collect all the paths of the run tree which correspond to the quantified path in the tree (recall that due to universal (ε-) transitions, there may be more than one path in the run tree that represent the same path in the original tree). Thus, for a successful run it needs to be checked that all the paths in a sequence of sets P are accepting. Since in a construction step we let P' be in the final ε-expansion of $\widehat{P}$, we obtain a situation illustrated as follows:



In order to check each path through the sets P, we employ an ω-word-automaton $\mathcal{C}$. This ω-word-automaton can nondeterministically guess a path through the sets P which violates the parity condition. In the theory of ω-automata, we can determinize and complement this ω-word-automaton (see e.g. [GTW02]) and then obtain a deterministic ω-word-automaton $\mathcal{D}$, which ensures that all paths through the P-sets fulfill the acceptance condition. We now let this ω-word-automaton $\mathcal{D}$ run in parallel with the ranked alternating tree automaton $\mathcal{B}$ on the quantified path and thus obtain a ranked alternating tree automaton $\mathcal{B}'$ with state set

$$Q^{\mathcal{B}'} = Q^{\mathcal{A}} \cup (\{q_0^{\mathcal{A}}\}, p_0^{\mathcal{D}}) \cup \{(P, p) \mid P \subseteq Q_\exists^{\mathcal{A}} \cup Q_\forall^{\mathcal{A}}, p \in Q^{\mathcal{D}}\}$$
$$\cup \{A^P \mid A \subseteq Q_\exists^{\mathcal{A}} \cup Q_\forall^{\mathcal{A}}, P \subseteq Q_\exists^{\mathcal{A}} \cup Q_\forall^{\mathcal{A}}\}$$

where for the first component $Q^{\mathcal{A}}$, we have any given kind of state, the second and third components consist of existential states of $Q_\exists$ only, and the fourth component consists of universal states of $Q^\varepsilon_\forall$ only.

The transition relation $\Delta^{\mathcal{B}'}$ consists of transitions of $\Delta^{\mathcal{A}_{00}}$, of ε-transitions starting in states A^P, as well as transitions of the form $((P,p)\,\overline{b},(P',p')\,\overline{c},A^{P'}\,0\,0)$ (restricted by $\mathcal{D}$-components p,p' to those transitions that fulfill the acceptance condition).

The acceptance component $\Omega^{\mathcal{B}'}$ is mainly inherited from the given automata of lower rank-sum, while for the states of highest rank-sum (which occur on the quantified path), the coloring of the states is determined by the ω-word-automaton $\mathcal{D}$.

The rank functions *rank* and *ε-rank* are again mainly adopted from the given automaton of lower rank-sum and defined as above for the states of highest rank-sum. ■

For the converse direction of Theorem 3.8 we have to supply a description of runs of ranked alternating automata in chain logic:

Theorem 3.13. *For any ranked alternating tree automaton $\mathcal{A}$ one can construct a formula φ of chain logic such that*

$$\mathcal{A} \textit{ accepts } \tau \Leftrightarrow \tau \models \varphi.$$

As a prerequisite, we define the relativization $\varphi(s)$ of a formula φ to a node u of a given tree τ such that $\tau, u \models \varphi(s) \Leftrightarrow \tau_{\downarrow u} \models \varphi$. This can be accomplished by relativizing the first-order quantifiers $\exists t(\ldots)$ to $\exists t(s \preceq t \wedge \ldots)$ such that we demand s to be a prefix of each node the formula talks about. This suffices to restrict the model of the formula to the subtree $\tau_{\downarrow u}$ at node u since we work with chain logic and properties of sets are expressed via quantification over first-order variables. Additionally, given a ranked alternating tree automaton $\mathcal{A}$ whose initial state q_0 has rank $n+1$, we may use formulas of chain logic for each of the automata $\mathcal{A}_p$ (obtained from $\mathcal{A}$ by declaring p of rank $\leq n$ as the initial state and restricting to those states reachable from p) denoted $\varphi_p(s)$. The formula holds for vertex u if the subtree $\tau_{\downarrow u}$ of the tree under consideration at vertex u is accepted by $\mathcal{A}_p$.

PROOF. We proceed by induction over the rank-sum $n+m$ of ranked alternating tree automata.

The induction starts with rank-sum 0, i.e., $\textit{rank}(\mathcal{A}) = \textit{ε-rank}(\mathcal{A}) = 0$. Clearly, the ranked alternating tree automata ACCEPT and REJECT can be described by the chain logic formulas $\varphi_{\text{ACCEPT}} = \forall s(s = s)$, resp. $\varphi_{\text{REJECT}} = \exists s(\neg s = s)$ (where each equality can be expressed via $\preceq$).

As the induction hypothesis, we assume that for a ranked alternating tree automaton $\mathcal{A}$ of rank-sum $n+m$, there exists a chain logic formula $\varphi_{\mathcal{A}}$ such that

$$\mathcal{A} \text{ has an accepting run on } \tau \Leftrightarrow \tau \models \varphi_{\mathcal{A}}.$$

For the induction step, we consider a ranked alternating tree automaton $\mathcal{B}$ of rank-sum $n+m+1$, and have to consider two cases:

1. We assume that the initial state q_0 of $\mathcal{B}$ is in $Q_\exists$. We omit treating the case $q_0 \in Q_\forall$ by taking the dual automaton $\widetilde{\mathcal{B}}$ for the complement (then we have that the initial state of $\widetilde{\mathcal{B}}$ is again in $Q_\exists$), and after the translation into a chain logic formula $\varphi_{\widetilde{\mathcal{B}}}$, we simply negate $\varphi_{\widetilde{\mathcal{B}}}$ and obtain the desired formula.

2. We assume that the initial state q_0 of $\mathcal{B}$ is in $Q^\varepsilon_\exists$. With the same method used above, we also cover the case $q_0 \in Q^\varepsilon_\forall$.

Case 1. We assume that $q_0 \in Q_\exists$ of rank-sum $n+m+1$. In an accepting run of $\mathcal{B}$ on tree τ, according to the format of ranked alternating tree automata, this implies that there is at most *one* path (finite or infinite) on which the highest rank is preserved. This is due to the fact that we start in an existential state and each switch to a universal state (or switches via ε-transitions) induces a decrease in ranks, thus we only have existential states on this path of highest rank. We will denote this path as set T_{q_0} illustrated in Figure 3.2. Thus, the existence of a successful run can be expressed as the existence of one partial run of highest rank-sum and other partial runs starting in states p of lower rank-sums (and using the transitions of $\mathcal{A}_p$, equivalent to φ_p as introduced above, resp.).

We call all states whose immediate predecessor is on the path T_{q_0} the *border* of T_{q_0}. The desired formula expresses:

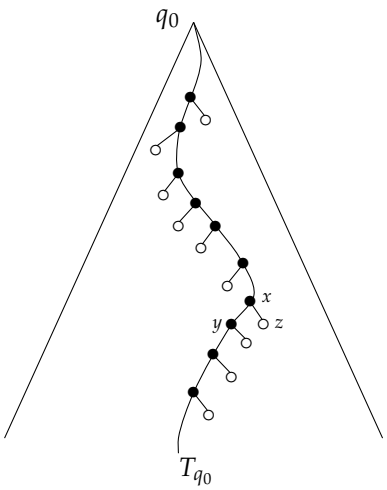
> "there is a finite or infinite path T_{q_0} such that T_{q_0} is labeled with states of the highest rank-sum, according to the transition function, such that for each vertex u on the border of T_{q_0}, if labeled with p, the subtree $\tau_{\downarrow u}$ is accepted by $\mathcal{A}_p$."

Let us fix some notation before tackling the formula $\varphi_{\mathcal{B}}$.

- By $\mathit{partition}(T_{q_0}, S^{q_1}_{q_0}, \ldots, S^{q_k}_{q_0})$ we denote that for the k states of same rank-sum as q_0, the disjoint sets $S^{q_1}_{q_0}, \ldots, S^{q_k}_{q_0}$ (representing the states) form a partition of the infinite set T_{q_0} and are compatible with the transitions of $\mathcal{B}$. It is not hard to see that this is expressible in chain logic.

- By $\mathit{partition}_{\mathit{fin}}(T_{q_0}, S^{q_1}_{q_0}, \ldots, S^{q_k}_{q_0})$ we denote that for the k states of same rank-sum as q_0, the disjoint sets $S^{q_1}_{q_0}, \ldots, S^{q_k}_{q_0}$ (representing the states) form a partition of the finite set T_{q_0} and are compatible with the transitions of $\mathcal{B}$.

Figure 3.2 The set T_{q_0} with its border nodes.



- As stated above, $border(T_{q_0})$ denotes the set of nodes u whose parent is in T_{q_0}, but where u itself is not (indicated in Figure 3.2 as circles).

- The formula $transition_0(x, y, z)$ states that there exists a transition such that the father node (represented by x) and the left son (denoted by y) are on the path of the highest rank-sum, and the right son (denoted by z) is not (and thus is in the border of T_{q_0} and is labeled with a state of lower rank-sum, this situation being indicated in Figure 3.2).

$$transition_0(x, y, z) = \bigvee_{(q\,a, \underline{q_1}\,a_1, q_2\,a_2) \in \Delta} \left(S^{q}_{q_0}(x) \wedge P_a(x) \wedge S^{q_1}_{q_0}(y) \wedge P_{a_1}(y) \wedge \varphi_{q_2}(z) \wedge P_{a_2}(z) \right)$$

  (where we indicate the rank preservation to the left by underlining)

- The formula $transition_1(x, y, z)$ is built analogously, indicating that the path of highest rank-sum goes through x and z.

$$transition_1(x, y, z) = \bigvee_{(q\,a, q_1\,a_1, \underline{q_2}\,a_2) \in \Delta} \left(S^{q}_{q_0}(x) \wedge P_a(x) \wedge S^{q_2}_{q_0}(z) \wedge P_{a_2}(z) \wedge \varphi_{q_1}(y) \wedge P_{a_1}(y) \right)$$

- The formula $transition_{fin}(x, y, z)$ covers the case that the set T_{q_0} is finite: it denotes that x (as the parent of left son y and right son z) is the last node in

T_{q_0}.

$$transition_{fin}(x,y,z) = \bigvee_{(q\,a,q_1\,a_1,q_2\,a_2)\in\Delta} \big(S^q_{q_0}(x) \wedge P_a(x) \wedge \varphi_{q_1}(y) \wedge P_{a_1}(y) \wedge \varphi_{q_2}(z) \wedge P_{a_2}(z)\big)$$

Now we are ready to define $\varphi_{\mathcal{B}}$:

$$\begin{aligned}
\varphi_{\mathcal{B}} := \exists T_{q_0} \exists S^{q_1}_{q_0} \cdots \exists S^{q_k}_{q_0} \Big[& \big(partition(T_{q_0}, S^{q_1}_{q_0}, \ldots, S^{q_k}_{q_0}) \\
& \wedge \forall z \in border(T_{q_0}) \forall x \forall y \Big(\big(Suc_0(x,y) \wedge Suc_1(x,z) \rightarrow transition_0(x,y,z)\big) \\
& \vee \big(Suc_0(x,z) \wedge Suc_1(x,y) \rightarrow transition_1(x,z,y)\big)\Big) \\
& \wedge \varphi_{acc}(T_{q_0})\big) \\
\vee \big(& partition_{fin}(T_{q_0}, S^{q_1}_{q_0}, \ldots, S^{q_k}_{q_0}) \wedge \forall z \in border(T_{q_0}) \forall x \forall y \Big(\\
& \big(Suc_0(x,y) \wedge Suc_1(x,z) \rightarrow transition_0(x,y,z)\big) \\
& \vee \big(Suc_0(x,z) \wedge Suc_1(x,y) \rightarrow transition_1(x,z,y)\big) \\
& \vee \big(y \in border(T_{q_0}) \wedge Suc_0(x,y) \wedge Suc_1(x,z) \rightarrow transition_{fin}(x,y,z)\big) \\
& \vee \big(y \in border(T_{q_0}) \wedge Suc_0(x,z) \wedge Suc_1(x,y) \rightarrow transition_{fin}(x,z,y)\big)\Big)\big)\Big]
\end{aligned}$$

The only item left to define is $\varphi_{acc}(T_{q_0})$ which expresses that on the infinite path T_{q_0}, the acceptance condition is fulfilled. Since we work with parity acceptance conditions, this is clearly formalizable in chain logic. When considering finite trees, it suffices to express the fulfillment of the acceptance condition at the end of the paths, i.e., at the leaves of the tree.

Case 2. We assume that $q_0 \in Q^{\varepsilon}_{\exists}$ is of rank-sum $n+m+1$. This can clearly be formalized in chain logic, as for an ε-transition, we immediately turn to states of a lower rank-sum, thus with the induction hypothesis, we obtain (with $\{q_0\,a_1 \rightarrow q^0_1, \ldots, q_0\,a_{k_0} \rightarrow q^0_{k_0}\}$ as the list of transitions for q_0):

$$\varphi_{\mathcal{B}} = \exists x \big(\neg \exists y (y \preceq x) \wedge \bigvee_{(q_0,a_j,q^0_j)\in\Delta} P_{a_j}(x) \wedge \varphi_{q_j}(x)\big)$$

■

Let us summarize the main result of this section:

Theorem 3.14 (Equivalence Theorem). *A tree language of infinite binary trees is recognizable by a ranked alternating tree automaton iff it is definable in chain logic, and both conversions, from automata to formulas and vice versa, are effective. The same equivalence holds if we refer to the domain of finite binary trees.*

We remark that the generalization from binary to finitely branching trees (over a ranked alphabet) is much more tedious in exposition but does not involve principal difficulties.

The Equivalence Theorem 3.14 provides an automata-theoretic characterization of an interesting fragment of monadic second-order logic. Let us point to another result of similar kind (probably the only one of this type), namely the result of Muller and Schupp [MSS86] that characterizes weak monadic second-order logic (where set quantification ranges over finite sets) in terms of weak alternating automata. In that paper the use of a rank function also appears, but it is of quite different nature since it is required to stay equal or decrease in *each* direction addressed in a transition. An earlier characterization of weak monadic second-order logic had been given by Rabin [Rab70], stating that a tree language T is definable in weak monadic second-order logic iff T and its complement are both recognized by nondeterministic Büchi tree automata.

In this thesis we do not enter possible applications of the equivalence result. Such applications may be pursued in two directions. First, the model of ranked alternating tree automaton might be useful in the algorithmic treatment of chain logic. Second, it provides an alternative track for showing results on the expressive power of chain logic, for example on separating chain logic from monadic second-order logic (as done in Thomas [Tho84] using model-theoretic methods). In both types of application, some technical features of ranked alternating tree automata prohibit an easy approach – in particular the interplay between alternation and ε-transitions – as it would be possible with standard nondeterministic tree automata.

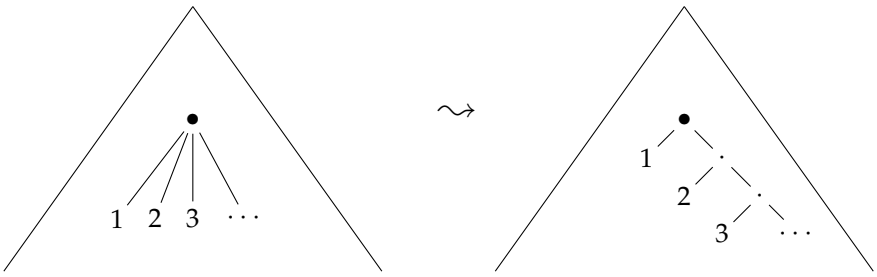
Chapter 4

Infinite-Branching Trees and Words over Infinite Alphabets

4.1 Motivation

The focus of the present chapter is the study of path logics over trees that are infinitely branching. We shall concentrate on the case where each tree node has an ω-sequence of children.

This case of infinitely branching labeled trees is already covered in the first landmark paper on automata over infinite trees, the work [Rab69] in which the decidability of the monadic second-order theory of the *binary* tree was shown. Rabin provided an MSO-interpretation of the ω-branching tree in the binary tree. The idea is simple: A point where infinite branching occurs is dissolved into a comb structure of the binary tree:



However, this interpretation involves a distortion of the tree levels. So, when we are dealing with extensions of path logic where, for instance, the binary equal-level predicate is adjoined, it is necessary to deviate from Rabin's approach.

A main motivation of the present chapter is to study path logic and chain logic over infinitely branching trees in the extension of the standard tree structure by the equal-level predicate. Our starting point is the result of Thomas [Tho90] that the chain logic theory of the binary tree extended by the equal-level predicate E is decidable. This is in contrast to the case where full MSO-logic is considered. In that case we obtain undecidability when the predicate E is adjoined; in fact, it is possible in this case to provide an interpretation of the infinite grid whose MSO-theory is undecidable (see [Tho90]).

The method of [Tho90] is a reduction to the MSO-theory of the successor structure $(\mathbb{N}, +1)$, in other words to the theory S1S. The idea is to code a chain of the

binary tree by a pair of two ω-words, one describing a path through the tree and the other indicating a subset of this path, i.e., a chain.

The generalization to infinite trees is possible when invoking a more expressive framework for the description of paths: Instead of the alphabet with the letters "go left" and "go right", a direction alphabet containing all natural numbers is now appropriate. So in order to prepare the case of infinite-branching trees, we have first to develop a theory of words and ω-words over infinite alphabets, an enterprise that is of value independently of the possible applications in the context of trees.

Another (and in fact more general) view of proceeding to infinitely branching trees is the construction of trees via tree iterations: A structure is unfolded into a tree. The case mentioned above originates from the tree iteration of $(\mathbb{N}, +1)$.

There already exists an extensive literature on languages (of finite or infinite words) over infinite alphabets, for example in the theory of "data words" (cf. e.g. [BDM+06]). An important motivation came from database theory (cf. e.g. [BFG05]), in particular in connection with the wish to cover XML-trees. In these trees, certain entries (labels) can be "data", for example natural numbers.

Additionally, in infinite-state verification it is interesting to be able to handle system runs where a single entry (state) is from an infinite set. Here the study of (infinite) words over an infinite alphabet may lead to a type of tree that still has labels in a finite alphabet but where infinite branching occurs: If one considers a Kripke structure (transition system) over an infinite domain of states (take again the example of natural numbers), the unfolding from a designated origin state yields an omega-branching tree.

We add new aspects to this theory by preparing several models of automaton and of logics over words from an infinite alphabet. We separate two issues: the logic that takes letter positions (i.e., natural numbers) for the underlying domain, and the logic that speaks about letter properties, i.e., which takes letters as elements of the domain.

For this, we shall work with an arbitrary relational structure $\mathcal{M}$ whose domain M serves as the alphabet, and a logic $\mathcal{L}$ by which we specify properties of letters from M (or – when we build words from the tuple alphabet M^n – properties of n-tuples from M). We then work with a standard logic such as MSO or FO to describe word models. The unary predicate expressions $P_a(x)$ (for "at position x there is letter a") of the classical approach are then replaced by predicates P_ψ where $\psi(x)$ or $\psi(x_1, \ldots, x_n)$ describes a letter property in the structure $\mathcal{M}$. The whole formalism is then denoted by terms such as $\mathcal{M}$-$\mathcal{L}$-MSO-logic. The pair $(\mathcal{M}, \mathcal{L})$ will be called an *alphabet frame*.

In the subsequent two sections this setting is introduced and basic results are shown which generalize work of Bès [Bès08]. The presence of infinite alphabets,

however, implies a weakness of the framework in the sense that now, for example, the equality of two successive letters in a word can not be expressed. In the classical approach we could use the following formula $\varphi(s,t)$ saying that at the successive positions s, t the letters agree:

$$Suc(s,t) \wedge \bigwedge_{a \in M} (P_a(s) \leftrightarrow P_a(t))$$

Over infinite alphabets this motivates to introduce the "clone predicate" C that is true of position s if the letter at s coincides with the previous letter. We shall consider and formally introduce several ways to take into account the aspect of "cloning" and related ideas.

The idea of cloning appeared earlier in the literature in the context of trees, namely in the construction of strong "tree iteration", first introduced by Muchnik (see [Sem84]) and given a full presentation by Walukiewicz [Wal02] and Courcelle and Walukiewicz [CW98]. We take the view here that it is useful to study this concept over the domain of word models in a separate investigation. This is done in the third section of this chapter.

In the final section we return to tree structures: the tree iterations in the context of chain logic. We start from the result of Thomas [Tho87] that the chain logic theory of the *binary* tree stays decidable when one adds the "equal-level predicate". We show that one can generalize this result to infinitely branching trees, using the results of Section 4.2.2, when the weak tree iteration of Shelah and Stupp is considered. On the other hand we prove that such a transfer fails for the strong tree iteration. The presented results are partially based on [STW11].

4.2 Languages over Structured Alphabets

4.2.1 Word Models: Definitions

The standard way of introducing words over finite alphabets as model-theoretic structures is based on using natural numbers for letter positions and using predicates indicating which letters are at which positions. To facilitate our technical considerations and to avoid additional considerations for special cases, we will only consider non-empty words.

Let $\Sigma = \{a_1, \ldots, a_k\}$ and $w = c_0 \cdots c_{\ell-1}$ be a word with $c_j \in \Sigma$ for $j = 0, \ldots, \ell - 1$. One represents w as a word model

$$\underline{w} = (\{0, \ldots, \ell - 1\}, Suc, <, P_{a_1}, \ldots, P_{a_k})$$

where $P_{a_i} = \{j \mid c_j = a_i\}$ for $0 \leq j \leq \ell - 1$. Similarly, models $\underline{\alpha}$ for ω-words α are introduced which only differ in allowing the universe to be $\mathbb{N}$.

Let us now turn to word models over infinite alphabets. There, the predicates P_a need to be replaced by predicates that are definable in an "alphabet logic" (which we will also refer to as "letter logic"). We consider alphabet symbols from a set M which is the universe of an *alphabet structure* $\mathcal{M}$. We shall work with relational alphabet structures:

$$\mathcal{M} = (M, R_1^M, \ldots, R_k^M),$$

where for technical reasons we require two distinguished elements, denoted 0 and 1, and represented by constants for which we also write 0, resp. 1. Call such a structure an *admissible alphabet structure*.

Typical examples are

- $(\mathbb{N}, Add, 0, 1)$ where Add is the graph of $+$,
- $(\mathbb{N}, \mathrm{Succ}, 0, 1)$,
- $(\mathbb{R}, <, 0, 1)$.

It is convenient to work with alphabets of the form M^n, i.e., to allow n-tuples over M as letters of "words over M". To emphasize the arity, we then sometimes speak of n-words.

The technical treatment below is simplified when we let an n-tuple $(\alpha_1, \ldots, \alpha_n)$ of ω-words over M be perceived as a single ω-word over M^n, the *convolution* of $(\alpha_1, \ldots, \alpha_n)$:

$$\langle \alpha_1, \ldots, \alpha_n \rangle := \begin{bmatrix} \alpha_1(0) \\ \vdots \\ \alpha_n(0) \end{bmatrix} \begin{bmatrix} \alpha_1(1) \\ \vdots \\ \alpha_n(1) \end{bmatrix} \cdots \quad \in (M^n)^\omega$$

Similarly, we define the *convolution of a relation* $R \subseteq (M^\omega)^n$ *of* ω*-words* to be the ω-language

$$L_R := \{\langle \alpha_1, \ldots, \alpha_n \rangle \mid (\alpha_1, \ldots, \alpha_n) \in R\} \subseteq (M^n)^\omega.$$

Analogous definitions can be given for the case of finite words over M^n. Here, an additional "padding" symbol $\notin M$ is used to fill the shorter words up to length. We omit the additional notations in order to increase readability.

In order to access letters from a set M^n in a formula talking about n-words, we have to introduce corresponding monadic predicates $P \subseteq M^n$. For a finite presentation of predicates, we use an "alphabet logic" $\mathcal{L}$ over an admissible alphabet structure $\mathcal{M}$. Any $\mathcal{L}$-formula $\psi(x_1, \ldots, x_n)$ defines the following predicate P_ψ over the domain of a word model:

$$P_\psi = \{(a_1, \ldots, a_n) \mid \mathcal{M} \models \psi[a_1, \ldots, a_n]\}$$

Thus, for a word $w \in (M^n)^+$ or $w \in (M^n)^\omega$ we introduce word models of the form

$$\underline{w} = (dom(w), +1, <, (P_\psi)_{\psi \in \Psi})$$

where Ψ is a (usually finite) set of $\mathcal{L}$-formulas $\psi(x_1, \ldots, x_n)$. For an n-word $w = \overline{a}_0 \cdots \overline{a}_{\ell-1}$ with $\overline{a}_i = (\overline{a}_i(1), \ldots, \overline{a}_i(n)) \in M^n$, we then have

$$P^{\underline{w}}_{\psi(x_1,\ldots,x_n)} = \{i \in dom(w) \mid \mathcal{M} \models \psi[\overline{a}_i(1), \ldots, \overline{a}_i(n)]\}.$$

We call the pair $(\mathcal{M}, \mathcal{L})$ consisting of an alphabet structure $\mathcal{M}$ and an alphabet logic $\mathcal{L}$ an *alphabet frame* and speak of n-words over the alphabet frame $(\mathcal{M}, \mathcal{L})$.

When speaking of a "logic $\mathcal{L}$", we mean any logic in the sense of abstract model theory [BF85, EFT07] which extends FO-logic and has an effective syntax with first-order variables. As standard examples, we mention monadic second-order logic, weak second-order logic, and transitive closure logic.

As standard example for an alphabet universe we shall use the set $\mathbb{N}$ of natural numbers (with the two distinguished elements 0 and 1). Natural examples of an alphabet frame $(\mathcal{M}, \mathcal{L})$ are the alphabet structures from above $((\mathbb{N}, Add, 0, 1)$, $(\mathbb{N}, \mathrm{Succ}, 0, 1)$, $(\mathbb{R}, <, 0, 1))$ with the logics $\mathcal{L} = $ MSO-logic, $\mathcal{L} = $ FO-logic.

In the consideration of the two logical systems that are chosen for the description of letter properties, resp. letter sequence properties (words), it is convenient to agree on separate notations. We shall use x, y, x_1, x_2, etc. for first-order variables of a logic $\mathcal{L}$ that speaks about letter properties, and we use s, t, t_1, etc. (reminding of "time") for letter positions in words. Similarly, we use corresponding capital letters X, Y, ... and S, T, ... as monadic second-order variables.

Cloned Word Models

As mentioned in the preamble of this chapter, in a word model representing an n-word over the alphabet frame $(\mathcal{M}, \mathcal{L})$, there is no direct mechanism to connect a letter from M^n to its neighboring letters in the word. There are several ways to introduce such a connection. In the sequel, two forms of establishing direct connections between successive letters are introduced.

Case (a) For an n-word model $\underline{w}$ we define C_i, a unary predicate of the word logic, as follows. Consider $w = \overline{a}_0 \cdots \overline{a}_{\ell-1}$ with

$$w = \begin{pmatrix} \overline{a}_0(1) \\ \vdots \\ \overline{a}_0(n) \end{pmatrix} \cdots \begin{pmatrix} \overline{a}_{\ell-1}(1) \\ \vdots \\ \overline{a}_{\ell-1}(n) \end{pmatrix}$$

and define C_i by

$$\underline{w} \models C_i[s] :\Leftrightarrow \overline{a}_s(i) = \overline{a}_{s-1}(i) \text{ for } s > 0.$$

In particular, C_i is always false for $s = 0$.

A more powerful construct offers the possibility of comparing letters at positions s, t which are further apart. We restrict here to the case of the alphabet $M = \mathbb{N}$. We introduce corresponding binary relations $\sim_i^d$ with the following semantics over words over the alphabet $\mathbb{N}^n$:

$$s \sim_i^d t :\Leftrightarrow \text{at positions } s, t, \text{ the } i\text{-th components have distance } d, \text{ i.e., } |\overline{a}_s(i) - \overline{a}_t(i)| = d$$

So the relation $s \sim_i^0 t$ may be termed "clone-on-distance relation". For $n = 1$ we cancel the index i.

Case (b) In a more general environment, we work with relations for letter pairs, i.e., predicates P connecting two word positions s, t. Over n-words, we allow predicates $P_{\psi(x_1,\ldots,x_n,y_1,\ldots,y_n)}(s,t)$ defined over

$$\underline{w} = \begin{pmatrix} \overline{a}_0(1) \\ \vdots \\ \overline{a}_0(n) \end{pmatrix} \cdots \begin{pmatrix} \overline{a}_{\ell-1}(1) \\ \vdots \\ \overline{a}_{\ell-1}(n) \end{pmatrix}$$

as follows: For $\overline{a}_s$ resp. $\overline{a}_t$ representing the letter at position s resp. t we have

$$\underline{w} \models P_{\psi(x_1,\ldots,x_n,y_1,\ldots,y_n)}(s,t) :\Leftrightarrow \mathcal{M} \models \psi[\overline{a}_s, \overline{a}_t].$$

Note that the special case $\psi : x_i = y_i$ defines C_i.

Projections The projection operation over n-words can be understood in several ways. Let us first consider the case of projecting words over M^{n+1} to words over M^n by deleting one component in the alphabet, referring to logical formulas of the alphabet logic $\mathcal{L}$.

In a letter predicate P_ψ, where $\psi = \psi(x_1, \ldots, x_{n+1})$, we would pass to P_φ where

$$\varphi(x_1, \ldots, x_n) := \exists x_{n+1} \psi(x_1, \ldots, x_{n+1}).$$

So an $(n+1)$-word model $\underline{w}$ over alphabet frame $(\mathcal{M}, \mathcal{L})$ is changed into an n-word model $proj_{[1,n]}(\underline{w})$. On the level of languages, $L \subseteq (M^{n+1})^+$ is transformed to

$$proj_{[1,n]}(L) := \{\overline{a}_0 \cdots \overline{a}_{\ell-1} \in (M^n)^+ \mid \exists f : \{0, \ldots, \ell-1\} \to M \text{ such that } \begin{pmatrix} \overline{a}_0 \\ f(0) \end{pmatrix} \cdots \begin{pmatrix} \overline{a}_{\ell-1} \\ f(\ell-1) \end{pmatrix} \in L\}$$

Thus an appropriate logic over words for this kind of projection would involve function quantifiers and thus surpass the expressive power of MSO-logic. We do not pursue this case in the present work.

In the sequel we deal with a special case of projection, namely "Boolean projection". At this point we use the fact that we concentrate on *admissible* alphabet structures in which two designated elements 0 and 1 exist. Here a letter predicate $P_\psi \subseteq M^n \times \{0,1\}$ is used with a formula $\psi(x_1,\ldots,x_{n+1})$ in which the condition $x_{n+1} = 0 \vee x_{n+1} = 1$ is built in.

Formally we work with formulas $\psi(x_1,\ldots,x_{n+1})$ that are equivalent to

$$\psi(x_1,\ldots,x_{n+1}) \wedge (x_{n+1} = 0 \vee x_{n+1} = 1).$$

Then the Boolean projection operation $pr_{[1,n]}$ transforms a language $L \subseteq (M^n \times \{0,1\})^+$ to

$$pr_{[1,n]}(L) := \{\overline{a}_0 \cdots \overline{a}_{\ell-1} \in (M^n)^+ \mid \exists f : \{0,\ldots,\ell-1\} \to \{0,1\} \text{ such that } \binom{\overline{a}_0}{f(0)} \cdots \binom{\overline{a}_{\ell-1}}{f(\ell-1)} \in L\}$$

This can be captured by a set quantifier:

$$pr_{[1,n]}(L) = \{\overline{a}_0 \cdots \overline{a}_{\ell-1} \in (M^n)^+ \mid \exists S \subseteq \{0,\ldots,\ell-1\} \text{ such that } \binom{\overline{a}_0}{\chi(0)} \cdots \binom{\overline{a}_{\ell-1}}{\chi(\ell-1)} \in L\}$$

where $\chi(i) = \begin{cases} 0 & i \notin S \\ 1 & i \in S \end{cases}$

As a third type of projection we consider 1-words and the transformation of letters by a definable function from M to M. Formally, we work with an alphabet frame $(\mathcal{M}, \mathcal{L})$ and consider a function $p : M \to M$ definable by an $\mathcal{L}$-formula $\psi_p(x,y)$:

$$\mathcal{M} \models \psi_p[a,a'] \Leftrightarrow p(a) = a'$$

The application of such a projection p, which we will refer to as "$\mathcal{L}$-definable projection", changes a word $a_0 \cdots a_{\ell-1} \in M^+$ to the word $p(a_0) \cdots p(a_{\ell-1}) \in M^+$.

Remark 4.1. If $L \subseteq M^+$ over the alphabet frame $(\mathcal{M}, \mathcal{L})$ is FO- or MSO-definable, say by the sentence φ and the projection $p : M \to M$ is $\mathcal{L}$-defined by ψ_p, then $p(L)$ is definable by the sentence φ' obtained from φ by replacing the predicate P_ψ where $\psi = \psi(x_1)$, with $P_{\psi'}$ where $\psi'(x_1) = \exists y(\psi_p(y) = x_1)$. ◀

Example 4.2. The function $p : \mathbb{N} \to \mathbb{N}$ with $p(n) =$ remainder of $n \mod 3$ is FO-definable over $(\mathbb{N}, +)$ by a formula $\psi(x,y)$ that expresses

$$\text{“}0 \leq y < 3\text{”} \wedge \text{“}x \mod 3 = y \mod 3\text{”}.$$

4.2.2 $(\mathcal{M}, \mathcal{L})$-Automata

In this section we introduce finite automata over finite words and Büchi automata over infinite words, following an approach that was introduced (for special cases) in the work of Bès [Bès08].

4.2.2.1 Definitions and Nonemptiness Problem

First, for a given alphabet frame $(\mathcal{M}, \mathcal{L})$, we will treat the automata definitions of $\mathcal{M}$-$\mathcal{L}$-automata for the case of finite words (as done in [Bès08]), in order to ease the notation for simple automata constructions. Languages accepted by these automata will be denoted as $\mathcal{M}$-$\mathcal{L}$-*recognizable* languages. We continue with the case of infinite words.

Definition 4.3. Let $(\mathcal{M}, \mathcal{L})$ be an alphabet frame where $\mathcal{M}$ is a structure with domain M. An $\mathcal{M}$-$\mathcal{L}$-automaton over *finite words* of n-tuples of M-elements is of the form

$$\mathcal{B} = (Q, M^n, q_0, \Delta, F)$$

where

- Q is a finite set of states,
- M^n is the input alphabet,
- $q_0 \in Q$ is the initial state,
- $\Delta \subseteq Q \times \Psi_n \times Q$ is the finite transition relation, where Ψ_n is the set of $\mathcal{L}$-formulas with n free variables,
- and $F \subseteq Q$ is the set of accepting states.

Given a tuple $(w_1, \ldots, w_n)$ of finite words over M, let $w = \langle w_1, \ldots, w_n \rangle$. Then a *run* of $\mathcal{B}$ on w is a finite sequence of states $\rho = \rho(0)\rho(1)\ldots\rho(m)$ with $\rho(0) = q_0$ such that for every $i \geq 0$ there exists an $\mathcal{M}$-$\mathcal{L}$-formula $\psi(x_1, \ldots, x_n)$ and a transition $(\rho(i), \psi, \rho(i+1))$ satisfying

$$\mathcal{M} \models \psi[w_1(i), \ldots, w_n(i)]$$

A run ρ of $\mathcal{B}$ on w is *successful* if $\rho(m) \in F$. We say that $\mathcal{B}$ *accepts* w if there exists a successful run of $\mathcal{B}$ on w. We denote by $L(\mathcal{B})$ the set of finite words over M^n accepted by $\mathcal{B}$.

Similarly, we proceed for Büchi automata:

Definition 4.4. Let $(\mathcal{M}, \mathcal{L})$ be an alphabet frame where $\mathcal{M}$ is a structure with domain M. An $\mathcal{M}$-$\mathcal{L}$-Büchi-automaton over n-tuples of M-elements is of the form

$$\mathcal{B} = (Q, M^n, q_0, \Delta, F)$$

where Q, M^n, q_0, Δ, and F are as in Definition 4.3, but acceptance is defined over ω-words:

If $\alpha = \langle \alpha_1, \ldots, \alpha_n \rangle$ is an ω-word over M^n, a *run* of $\mathcal{B}$ on α is an infinite sequence of states $\rho = \rho(0)\rho(1)\ldots$ with $\rho(0) = q_0$ such that for every $i \geq 0$ there exists an $\mathcal{M}$-$\mathcal{L}$-formula $\psi(x_1, \ldots, x_n)$ and a transition $(\rho(i), \psi, \rho(i+1))$ satisfying

$$\mathcal{M} \models \psi[\alpha_1(i), \ldots, \alpha_n(i)].$$

A run ρ of $\mathcal{B}$ on α is *successful* if there exist infinitely many i such that $\rho(i) \in F$. We say that $\mathcal{B}$ *accepts* α if there exists a successful run of $\mathcal{B}$ on α. We denote by $L(\mathcal{B})$ the set of ω-words over M^n accepted by $\mathcal{B}$.

Remark 4.5. We introduce $\mathcal{M}$-$\mathcal{L}$-Büchi automata for dimensions $n \geq 0$. In certain settings, the dimensions will need to be restricted in order to keep decidability. If there is no explicit remark on the dimensions, we always refer to dimensions $n \geq 0$. ◂

We start by showing decidability of the nonemptiness problem.

Proposition 4.6. *If the $\mathcal{L}$-theory of $\mathcal{M}$ is decidable, then the nonemptiness problem for $\mathcal{M}$-$\mathcal{L}$-automata on finite words as well as for $\mathcal{M}$-$\mathcal{L}$-Büchi automata is decidable.*

PROOF. For both kinds of $\mathcal{M}$-$\mathcal{L}$-automata, we have to determine whether there exists a word which is the label of a successful run. Unlike in the classical case, the existence of a transition $(p, \psi, q) \in \Delta$ does not necessarily imply the existence of an input letter $\bar{a} \in M^n$ that satisfies ψ. Thus as a preparation, we have to check for each of the finitely many transitions $(p, \psi(x_1, \ldots, x_n), q) \in \Delta$ whether it is "useful", i.e., whether there is an input letter $\bar{a} \in M^n$ satisfying ψ. This is done by invoking decidability of the $\mathcal{L}$-theory of $\mathcal{M}$, namely by checking whether $\mathcal{M} \models \exists x_1 \ldots \exists x_n \psi(x_1, \ldots, x_n)$. Now one considers the directed graph (Q, R) where $(p, q) \in R$ if there is a useful transition from p to q. For an $\mathcal{M}$-$\mathcal{L}$-automaton over finite words, it remains to check whether in (Q, R) there is a path from q_0 to F; for an $\mathcal{M}$-$\mathcal{L}$-Büchi automaton one verifies whether in (Q, R) there is a path from q_0 to a strongly connected component containing a state from F. ■

Next we turn to the basic closure properties.

In all the considerations to follow below, we work with alphabet frame $(\mathcal{M}, \mathcal{L})$ such that the $\mathcal{L}$-theory of $\mathcal{M}$ is decidable. We call such alphabet frames *effective*. This assumption allows us to determine which transitions (p, ψ, q) are in fact "executable".

4.2.2.2 Closure Properties

We want to develop basic closure properties over $\mathcal{M}$-$\mathcal{L}$-recognizable languages. Recall that these are languages over finite words (over infinite alphabets).

With these closure properties, we prepare for a theory of $\mathcal{M}$-$\mathcal{L}$-recognizable languages and therewith lay the foundation for a theory over infinite words; with the aim to show the equivalence of $\mathcal{M}$-$\mathcal{L}$-(Büchi)-recognizability and $\mathcal{M}$-$\mathcal{L}$-MSO-definability, which will fully be addressed in Section 4.2.2.3.

We fix an alphabet frame $(\mathcal{M}, \mathcal{L})$ and start with a preparation and show that nondeterministic and deterministic $\mathcal{M}$-$\mathcal{L}$-automata are of the same expressive power. We establish the equivalence by exploiting the well-known idea of the powerset construction. However, we need to adapt the classical construction in order to deal with the $\mathcal{L}$-formulas in the transitions of the automata.

Determinization

Proposition 4.7. *For an effective alphabet frame $(\mathcal{M}, \mathcal{L})$, a nondeterministic $\mathcal{M}$-$\mathcal{L}$-automaton can be transformed into an equivalent deterministic $\mathcal{M}$-$\mathcal{L}$-automaton.*

PROOF. As mentioned above, we need more effort for the determinization of a nondeterministic $\mathcal{M}$-$\mathcal{L}$-automaton $\mathcal{B} = (Q, M^n, q_0, \Delta, F)$ compared to the classical determinization of finite automata (over finite alphabets). We follow the approach roughly sketched by Bès [Bès08] but present a proof in some more detail and refer to an arbitrary alphabet frame. Given an $\mathcal{M}$-$\mathcal{L}$-automaton $\mathcal{B}$ (on finite words), $\mathcal{B}$ does not necessarily provide a run (accepting or not accepting) for every possible input letter in M^n, i.e., there might be a letter that does not satisfy any of the formulas of the applicable transitions. Thus, we modify the set of formulas for the transitions such that each input word leads to a complete run, and additionally, we prepare for determinism. The construction is divided into two parts: First, we need to find a suitable set Ψ of formulas that we can use as transition labels, and subsequently, we apply a powerset construction similar to the well-known powerset construction for nondeterministic automata on $\mathcal{B}$.

In the first step, we define a set Ψ of formulas which we can use as transition labels resulting in an automaton $\mathcal{B}' = (Q, M^n, q_0, \Delta', F)$ with $L(\mathcal{B}') = L(\mathcal{B})$. In order to prepare for determinism, the set Ψ of formulas has to fulfill two properties:

(a) No two formulas of the set Ψ can be satisfied by the same symbol. This yields unique runs for each word in $(M^n)^*$.

(b) The set Ψ of formulas is complete, i.e., in each state of the automaton and for each letter $\overline{a} \in M^n$ there exists a transition labeled by a formula ψ such that $\mathcal{M} \models \psi[\overline{a}]$.

The desired set Ψ of formulas can be constructed as follows: Let $\psi_1, \ldots, \psi_k$ denote the formulas which appear in the transitions of the nondeterministic $\mathcal{B}$. We introduce for each subset $J \subseteq \{1, \ldots, k\}$, the formula $\chi_J := \bigwedge_{i \in J} \psi_i \wedge \bigwedge_{i \notin J} \neg\psi_i$. Then, our new set of formulas is $\Psi' = \{\chi_J \mid J \subseteq \{1, \ldots, k\}\}$. Indeed, the demanded properties (a) and (b) are fulfilled:

1. Consider two different sets $J, K \subseteq \{1, \ldots, k\}$ with $J \neq K$ and corresponding formulas χ_J and χ_K. Assume towards a contradiction that there exists a symbol $\overline{a} \in M^n$ such that $\mathcal{M} \models \chi_J[\overline{a}]$ and $\mathcal{M} \models \chi_K[\overline{a}]$. Then, $\mathcal{M} \models \chi_J \wedge \chi_K[\overline{a}]$. Due to the construction of the formulas, they differ at least in one member of the conjunction, say ψ_i is a subformula of χ_J and $\neg\psi_i$ is a subformula of χ_J. Hence, we have $\mathcal{M} \models \psi_i \wedge \neg\psi_i$ which is a contradiction. Thus, no two formulas of Ψ' can be satisfied by the same symbol.

2. First, consider the case that there already exists a given formula ψ_i in $\mathcal{B}$ such that a symbol $\overline{a} \in M^n$ satisfies ψ_i. Then, by construction of the formulas χ_J there exists a Boolean combination χ_J, $i \in J$ of all formulas $\psi_1, \ldots, \psi_k$ such that $\overline{a}$ satisfies χ_J. Conversely, consider the case that a symbol $\overline{a} \in M^n$ satisfies none of the existing formulas ψ_i, then $\overline{a}$ satisfies the formula χ_J with $J = \emptyset$ which is a conjunction of all $\neg\psi_i$.

The next steps are to replace the transitions in $\mathcal{B}$, such that the transition labels consist of the formulas χ_J of Ψ' and hereafter apply a powerset construction, similar to the classical case, which takes advantage of the newly introduced formulas. We replace every transition $(p, \psi_i, q) \in \Delta$ with all transitions of the form (p, Ψ_i, q) with $\Psi_i = \bigvee_{J \ni i} \chi_J$ and thus obtain $\mathcal{B}'$ with a new transition relation Δ'. Let us now show, that $\mathcal{B}'$ still recognizes $L(\mathcal{B})$. For this we show that a transition from p to q in $\mathcal{B}'$ is executable via a symbol $\overline{a}$ if and only if there exists a transition from p to q in $\mathcal{B}$ that is executable via $\overline{a}$. There are two cases we can distinguish:

1. There exists a symbol $\overline{a} \in M^n$ such that the transition $(p, \psi_i, q) \in \Delta$ of the unmodified automaton $\mathcal{B}$ is executable via $\overline{a}$. Then $\mathcal{M} \models \psi_i[\overline{a}]$, and by construction there exists a formula χ_J where $i \in J \subseteq \{1, \ldots, k\}$ such that $\mathcal{M} \models \chi_J[\overline{a}]$, and $(p, \Psi_i, q) \in \Delta'$ where Ψ_i is a disjunction over all sets J that contain i.

2. There exists a symbol $\overline{a} \in M^n$ such that the transition $(p, \psi_i, q) \in \Delta$ of the unmodified automaton $\mathcal{B}$ is not executable via $\overline{a}$. Then $\mathcal{M} \not\models \psi_i[\overline{a}]$, and by construction every transition of the form $(p, \Psi_i, q) \in \Delta'$ with $\Psi_i = \bigvee_{J \ni i} \chi_J$ is not executable, because ψ_i is a subformula of each χ_J.

Thus, $L(\mathcal{B}') = L(\mathcal{B})$.

Now we apply a powerset construction as follows. As above, let $\psi_1, \ldots, \psi_k$ denote the formulas which appear in the transitions of $\mathcal{B}$. Consider, for every subset

$J \subseteq \{1, \ldots, k\}$, the formula $\chi_J := \bigwedge_{i \in J} \psi_i \wedge \bigwedge_{i \notin J} \neg \psi_i$. With $\mathcal{B}' = (Q, M^n, q_0, \Delta', F)$ as the $\mathcal{M}$-$\mathcal{L}$-automaton modified as described in the second step above, we obtain the $\mathcal{M}$-$\mathcal{L}$-automaton $\mathcal{B}'' = (Q'', M^n, q_0'', \Delta'', F'')$, where

- $Q'' = \mathcal{P}(Q)$ (powerset of Q),
- $q_0'' = \{q_0\}$,
- $\Delta'' \subseteq Q'' \times \Psi_n \times Q''$, where Ψ_n is the set of $\mathcal{L}$-formulas with n free variables in which for each $P \subseteq Q$ and $J \subseteq \{1, \ldots, k\}$: (P, Ψ_i, S) with $S = \{s \mid p \in P : (p, \Psi_i, s) \in \Delta'\}$,
- $F'' = \{P \subseteq Q \mid P \cap F \neq \emptyset\}$.

In the resulting deterministic $\mathcal{M}$-$\mathcal{L}$-automaton $\mathcal{B}''$ for every state q and every formula Ψ_i there exists a single outgoing transition labeled Ψ_i.

We show the equivalence between the $\mathcal{M}$-$\mathcal{L}$-automaton $\mathcal{B}'$ and the deterministic $\mathcal{M}$-$\mathcal{L}$-automaton $\mathcal{B}''$.

Given a tuple $(w_1, \ldots, w_n)$ of finite words over M, we show that

$$\mathcal{B}' : q_0 \xrightarrow{\langle w_1, \ldots, w_n \rangle} q \text{ if and only if } \mathcal{B}'' : \{q_0\} \xrightarrow{\langle w_1, \ldots, w_n \rangle} S,\ q \in S$$

by induction over the length of $\langle w_1, \ldots, w_n \rangle$.

If $|\langle w_1, \ldots, w_n \rangle| = 0$, then $\langle w_1, \ldots, w_n \rangle = \varepsilon$ and no transition is executed in $\mathcal{B}''$ nor in $\mathcal{B}'$ and both automata stay in their initial states, and obviously $q_0 \in \{q_0\}$.

For the induction step consider $|\langle w_1, \ldots, w_n \rangle| = \ell$ with $\langle w_1, \ldots, w_n \rangle = \overline{u}\overline{a}$ with $\overline{a} \in M^n$ and $\overline{u}$ is a word over M^n. In $\mathcal{B}'$ exists a run $q_0 \xrightarrow{\langle w_1, \ldots, w_n \rangle} q$, i.e. a run $q_0 \xrightarrow{\overline{u}\overline{a}} q$. Then there exists a state $p \in Q$ such that $\mathcal{B}' : q_0 \xrightarrow{\overline{u}} p$ and $\mathcal{B}' : p \xrightarrow{\overline{a}} q$. By induction hypothesis, there exists a run $\mathcal{B}'' : \{q_0\} \xrightarrow{\overline{u}} P,\ p \in P$. The existence of a run $p \xrightarrow{\overline{a}} q$ in $\mathcal{B}'$ implies the existence of a transition of the from $(p, \Psi_i, q) \in \Delta'$ of $\mathcal{B}'$. By construction of Δ'' of $\mathcal{B}''$ there exists a transition $(P, \Psi_i, S) \in \Delta''$ with $q \in S$, because $(p, \Psi_i, q) \in \Delta'$. Thus, there exists a run $\mathcal{B}'' : P \xrightarrow{\overline{a}} S$. All in all, we obtain $\mathcal{B}'' : \{q_0\} \xrightarrow{\overline{u}} P \xrightarrow{\overline{a}} S$. Therefore, there exists a run $\mathcal{B}'' : \{q_0\} \xrightarrow{\overline{u}\overline{a}} S$ that is a run $\mathcal{B}'' : \{q_0\} \xrightarrow{\langle w_1, \ldots, w_n \rangle} S,\ q \in S$.

What is left is to prove that $L(\mathcal{B}') = L(\mathcal{B}'')$. Therefore, we show for all tuples $(w_1, \ldots, w_n)$ of words: $\langle w_1, \ldots, w_n \rangle \in L(\mathcal{B}')$ iff $\langle w_1, \ldots, w_n \rangle \in L(\mathcal{B}'')$:

$$\begin{aligned}
\langle w_1, \ldots, w_n \rangle \in L(\mathcal{B}') \quad & \text{iff } \exists q\ \mathcal{B}' : q_0 \xrightarrow{\langle w_1, \ldots, w_n \rangle} q \text{ and } q \in F \\
& \text{iff } \mathcal{B}'' : \{q_0\} \xrightarrow{\langle w_1, \ldots, w_n \rangle} S,\ q \in S \text{ and } S \in F'', \\
& \quad \text{because } S \cap F \neq \emptyset \\
& \text{iff } \langle w_1, \ldots, w_n \rangle \in L(\mathcal{B}'')
\end{aligned}$$

■

With this preparation, we are now able to show the desired closure properties in a straightforward way.

Lemma 4.8. *For an effective alphabet frame $(\mathcal{M}, \mathcal{L})$, the class of $\mathcal{M}$-$\mathcal{L}$-recognizable languages (of finite n-words) is closed under union, projection from M^{n+1} to M^n, and complementation.*

PROOF. The closure properties of $\mathcal{M}$-$\mathcal{L}$-recognizable languages (of finite words) are shown by slight adaptions of the classical case (where the alphabet is finite). Thus, we will resort to the classical constructions and provide rough outlines of analogous constructions only, omitting the easy correctness proofs.

The closure under union is completely analogous to the classical construction for nondeterministic finite automata over finite alphabets. For two $\mathcal{M}$-$\mathcal{L}$-automata $\mathcal{B}_i = (Q^i, M^n, q_0^i, \Delta^i, F^i)$ for $i \in \{1,2\}$, we construct the union $\mathcal{M}$-$\mathcal{L}$-automaton $\mathcal{B} = (Q, M^n, q_0, \Delta, F)$ where $Q = Q_1 \uplus Q_2 \uplus \{q_0\}$, $\Delta = \Delta_1 \cup \Delta_2 \cup \{(q_0, \psi, q) \mid (q_0^1, \psi, q) \in \Delta^1 \text{ or } (q_0^2, \psi, q) \in \Delta^2\}$, and $F = F^1 \cup F^2$.

An automaton for the projection from M^n to M^{n-1} can easily be obtained by replacing the "label" $\psi(x_1, \ldots, x_n)$ of a transition by $\exists x_n \psi(x_1, \ldots, x_n)$.

For the complementation, we follow the strategy to determinize the given automaton and then swapping final states with nonfinal states. With the preparation given in Proposition 4.7, we construct for a given $\mathcal{M}$-$\mathcal{L}$-automaton $\mathcal{B}$ the equivalent deterministic $\mathcal{M}$-$\mathcal{L}$-automaton $\mathcal{B}''$ and then simply swap the sets F'' and $Q'' \setminus F''$. ∎

After treating the basic closure properties for languages over finite words, we now start the transition to languages over infinite words and their basic closure properties. We will treat some useful constructions first, which will be used later on to facilitate the more involved constructions.

Lemma 4.9. *Given an effective alphabet frame $(\mathcal{M}, \mathcal{L})$, for an $\mathcal{M}$-$\mathcal{L}$-recognizable language (of finite words) $U \subseteq (M^n)^*$ and an $\mathcal{M}$-$\mathcal{L}$-Büchi recognizable ω-language $K \subseteq (M^n)^\omega$, we have*

1. *U^ω is $\mathcal{M}$-$\mathcal{L}$-Büchi recognizable,*
2. *$U \cdot K$ is $\mathcal{M}$-$\mathcal{L}$-Büchi recognizable,*

and the construction of these Büchi automata is effective.

PROOF. Concerning the first part of the Lemma, for a given $\mathcal{M}$-$\mathcal{L}$-recognizable $U \subseteq (M^n)^*$, the construction of an $\mathcal{M}$-$\mathcal{L}$-Büchi automaton recognizing U^ω can be done in a straightforward way by isolating the initial state such that it has no incoming transitions and for each transition from a state q to some state in F, adding a transition from q to the new initial state over the same letter (while respecting the roles of the old and new initial state), where the new initial state will be the only final state in the new automaton: Let $\mathcal{M}$-$\mathcal{L}$-automaton $\mathcal{B} =$

(Q, M^n, q_0, Δ, F) recognize U. Then, we construct the $\mathcal{M}$-$\mathcal{L}$-Büchi automaton $\mathcal{B}' = (Q \cup \{q_0'\}, M^n, q_0', \Delta', \{q_0'\})$ with $\Delta' = \Delta \cup \{(q_0', \psi, q) \mid (q_0, \psi, q) \in \Delta\} \cup \{(q, \psi, q_0') \mid (q, \psi, p) \in \Delta \wedge p \in F\} \cup \{(q_0', \psi, q_0') \mid (q_0, \psi, p) \in \Delta \wedge p \in F\}$.

For the concatenation $U \cdot K$, we again follow a well-known idea by composing the two automata with additional transitions to cross over from one to the other at the appropriate positions: This means, that for given $\mathcal{M}$-$\mathcal{L}$-automaton $\mathcal{B}_1 = (Q^1, M^n, q_0^1, \Delta^1, F^1)$ recognizing $U \subseteq (M^n)^*$ and $\mathcal{M}$-$\mathcal{L}$-Büchi automaton $\mathcal{B}_2 = (Q^2, M^n, q_0^2, \Delta^2, F^2)$ with $L(\mathcal{B}_2) = K \subseteq (M^n)^\omega$, we ensure that for every transition leading to a final state in $\mathcal{B}_1$ we introduce an additional transition leading to q_0^2 in $\mathcal{B}_2$, and for the case that $\varepsilon \in L(\mathcal{B}_1)$, i.e. $q_0^1 \in F^1$, we enable the resulting automaton to start immediately with a word in $L(\mathcal{B}_2)$. This yields $\mathcal{C} = (Q^1 \cup Q^2, M^n, q_0^1, \Delta, F_2)$ with $\Delta = \Delta^1 \cup \Delta^2 \cup \{(p, \psi, q_0^2) \mid (p, \psi, q) \in \Delta^1 \wedge p \in F^1\} \cup \{(q_0^1, \psi, q) \mid \text{ if } q_0^1 \in F^1 \wedge (q_0^2, \psi, q) \in \Delta^2\}$. ∎

Lemma 4.10. *Over an effective alphabet frame $(\mathcal{M}, \mathcal{L})$, the class of $\mathcal{M}$-$\mathcal{L}$-Büchi-recognizable ω-languages is closed under union and projection (from M^n to M^{n-1}).*

PROOF. For union and projection, the constructions are analogous to the case of $\mathcal{M}$-$\mathcal{L}$-automata over finite words (as given in Lemma 4.8). By the obvious adaption of the constructions for the classical case, we obtain the respective $\mathcal{M}$-$\mathcal{L}$-Büchi automata.

Hence, for the closure under union, with given $\mathcal{M}$-$\mathcal{L}$-Büchi automata $\mathcal{B}_i = (Q^i, M^n, q_0^i, \Delta^i, F^i)$ for $i \in \{1, 2\}$, we construct the union $\mathcal{M}$-$\mathcal{L}$-Büchi automaton $\mathcal{B} = (Q_1 \cup Q_2 \cup \{q_0\}, M^n, q_0, \Delta, F^1 \cup F^2)$ where $\Delta = \Delta_1 \cup \Delta_2 \cup \{(q_0, \psi, q) \mid (q_0^1, \psi, q) \in \Delta^1 \text{ or } (q_0^2, \psi, q) \in \Delta^2\}$.

For the projection from M^n to M^{n-1}, again replacing the label $\psi(x_1, \ldots, x_n)$ of a transition by $\exists x_n \psi(x_1, \ldots, x_n)$ suffices. ∎

Let us turn to the most interesting closure property of Büchi automata: complementation.

Lemma 4.11. *If the $\mathcal{L}$-theory of $\mathcal{M}$ is decidable, the class of $\mathcal{M}$-$\mathcal{L}$-Büchi-recognizable ω-languages is effectively closed under complementation.*

PROOF. We sketch the construction for complementation, using the original approach of Büchi [Büc62], which was worked out in [SVW87].

Let $\mathcal{B} = (Q, M^n, q_0, \Delta, F)$ be an $\mathcal{M}$-$\mathcal{L}$-Büchi automaton. We introduce an equivalence relation over finite M^n-words such that the complement of the recognized language $(M^n)^\omega \setminus L(\mathcal{B})$ is representable as a finite union of sets $U \cdot V^\omega$ with $\mathcal{M}$-$\mathcal{L}$-recognizable sets $U, V \subseteq (M^n)^*$. By Lemmas 4.9 and 4.10, this suffices to show $\mathcal{M}$-$\mathcal{L}$-Büchi recognizability of $(M^n)^\omega \setminus L(\mathcal{B})$.

The desired equivalence relation is defined in terms of *transition profiles*. The finite set of all transition profiles of a given automaton $\mathcal{B}$ completely determines

the behavior of $\mathcal{B}$ on a finite word. We will show that any infinite word can be divided into a sequence of only finitely many transition profiles (in fact, we will show that two suffice) which enables us to assess the behavior of $\mathcal{B}$ on an infinite word by means of the transition profiles only. Exploiting these transition profiles then allows us to derive the desired representation for the complement of $L(\mathcal{B})$.

We write for a finite word $u \in (M^n)^*$ and $p, q \in Q$:

- $\mathcal{B} : p \xrightarrow{u} q$ if there is a run on u from p to q in $\mathcal{B}$,
- $\mathcal{B} : p \xrightarrow[F]{u} q$ if there is a run on u from p to q in $\mathcal{B}$ that visits an accepting state from F.

A transition profile $\vartheta = tp(u)$ is then given by two sets $I_{tp(u)}$, $J_{tp(u)}$ of pairs of states, $I_{tp(u)}$ containing those pairs (p,q) where $\mathcal{B} : p \xrightarrow{u} q$, and $J_{tp(u)}$ containing those pairs (p,q) where $\mathcal{B} : p \xrightarrow[F]{u} q$. Two words u, v are called $\mathcal{B}$-equivalent, written $u \sim_{\mathcal{B}} v$, if $tp(u) = tp(v)$. This equivalence relation is of finite index: For this, note that each equivalence class (i.e., a language $U_\vartheta = \{u \mid tp(u) = \vartheta\}$ for a transition profile ϑ) is a Boolean combination of the $\mathcal{M}$-$\mathcal{L}$-recognizable languages $U_{pq} = \{u \mid \mathcal{B} : p \xrightarrow{u} q\}$, $U'_{pq} = \{u \mid \mathcal{B} : p \xrightarrow[F]{u} q\}$, in fact, we have

$$U_\vartheta = \bigcap_{(p,q)\in I_\vartheta} U_{pq} \cap \bigcap_{(p,q)\notin I_\vartheta} \overline{U_{pq}} \cap \bigcap_{(p,q)\in J_\vartheta} U'_{pq} \cap \bigcap_{(p,q)\notin J_\vartheta} \overline{U'_{pq}}.$$

Since the set of pairs (p,q) is finite, we get only finitely many equivalence classes.

Furthermore, by Lemma 4.8 and Proposition 4.6, we can compute those U_ϑ which are nonempty and hence obtain an effective presentation of the equivalence classes in terms of the corresponding finite sets I_ϑ, J_ϑ.

We identify the equivalence classes with the transition profiles and denote the set of these transition profiles of $\mathcal{B}$ by $TP_{\mathcal{B}}$.

The following "saturation property" is now immediate:

Lemma 4.12. *For any $\sim_{\mathcal{B}}$-equivalence classes U, V, the ω-language $U \cdot V^\omega$ is either contained in $L(\mathcal{B})$ or in its complement.*

It remains to show that any ω-word over M^n belongs to some set $U \cdot V^\omega$ where U, V are $\sim_{\mathcal{B}}$-classes.

For this we use the transition profiles as "colors" of segments $\alpha[i,j]$ for $i, j \in \mathbb{N}$ and invoke Ramsey's Infinity Lemma [Ram30]:

Lemma 4.13 (Ramsey [Ram30]). *Let X be a countably infinite set such that each undirected edge $\{x, y\}$ over X has a color from a finite set C of colors. Then there is an infinite subset Y of X such that all edges over Y have the same color.*

If we consider $\mathbb{N}$ as set X, there is an infinite subset $I = \{i_0 < i_1 < i_2 < \ldots\}$ corresponding to Ramsey's Lemma with $TP_{\mathcal{B}}$ as colors, such that there is for any

α and any $\mathcal{M}$-$\mathcal{L}$-Büchi automaton $\mathcal{B}$ a pair of transition profiles ϑ_0, ϑ from $TP_{\mathcal{B}}$ with

$$tp(\alpha[0, i_0 - 1]) = \vartheta_0, \;\; tp(\alpha[i_j, i_{j+1} - 1]) = \vartheta \text{ for } j \geq 0.$$

This shows that $\alpha \in U_{\vartheta_0} \cdot U_{\vartheta}^{\omega}$, where U_{ϑ_0}, U_{ϑ} denote the equivalence classes of $\sim_{\mathcal{B}}$ corresponding to ϑ_0 resp. ϑ. Let

$$NTP_{\mathcal{B}} = \{(\vartheta_0, \vartheta) \in TP_{\mathcal{B}}^2 \mid U_{\vartheta_0} \cdot U_{\vartheta}^{\omega} \cap L(\mathcal{B}) = \emptyset\}.$$

Again, by decidability of the $\mathcal{L}$-theory of $\mathcal{M}$, this set is computable. Then

$$(M^n)^{\omega} \setminus L(\mathcal{B}) = \bigcup_{(\vartheta_0, \vartheta) \in NTP_{\mathcal{B}}} U_{\vartheta_0} U_{\vartheta}^{\omega}.$$

■

4.2.2.3 Equivalence between MSO and Automata over $(\mathcal{M}, \mathcal{L})$

In this section, we establish the connection between automata over $(\mathcal{M}, \mathcal{L})$ and monadic second-order logic (MSO). Our main goal is to show that there is a general equivalence between $\mathcal{M}$-$\mathcal{L}$-recognizable languages (both of finite words and infinite words) over infinite alphabets and MSO-definability over $(\mathcal{M}, \mathcal{L})$.

Regarding finite alphabets, a general equivalence between languages of finite words recognized by finite automata and monadic second-order logic was found in the early 1960s. Büchi [Büc60], Elgot [Elg61], and Trakhtenbrot [Tra61] showed that MSO-definable languages describe exactly the class of regular languages; a full proof hereof can be found in [Tho97]. In [Büc62], Büchi showed a similar equivalence of automata over infinite words over a finite alphabet and monadic second-order logic over such words.

Thus, to establish such an equivalence, let us first introduce monadic second-order logic over words over $(\mathcal{M}, \mathcal{L})$. In order to characterize a language over a structure $\mathcal{M}$ by a logical formalism, we will combine two logics. In our scenario, the logic MSO is suitable to describe relations between positions in a word, e.g. "position s occurs before position t" or (for finite words) "the word is of even length". In order to express properties that letters on the positions have, MSO provides predicates, e.g. predicate P_a collects all positions that hold the letter a of the given alphabet, realized by $P_a(s)$ evaluating to true for each position s where letter a occurs. In our setting, we replace a collection of such predicates by making use of the alphabet logic $\mathcal{L}$. This allows us, given alphabet structure $\mathcal{M}$ with universe M and the logic $\mathcal{L}$, to describe properties of letters using $\mathcal{M}$-$\mathcal{L}$-formulas. Therewith, we introduce for any $\mathcal{M}$-$\mathcal{L}$-formula ψ a new predicate that expresses "the letter at position s has the property defined by the $\mathcal{M}$-$\mathcal{L}$-formula ψ", i.e., the symbol at position s satisfies ψ in $\mathcal{M}$. Thus we combine the logics

MSO and $\mathcal{L}$ by adding to the MSO formalism unary predicates P_ψ for every $\mathcal{M}$-$\mathcal{L}$-formula ψ. We call this new logical formalism $\mathcal{M}$-$\mathcal{L}$-MSO.

Definition 4.14. Given a structure $\mathcal{M}$ with domain M and a logic $\mathcal{L}$. We define $\mathcal{M}$-$\mathcal{L}$-MSO as MSO-logic with signature $\sigma = (<, Suc, (P_\psi)_{\psi \in \mathcal{M}\text{-}\mathcal{L}})$, where

- $<$ is the natural ordering relation symbol,
- Suc is the successor relation symbol, and
- P_ψ is a unary relation symbol associated to every $\mathcal{M}$-$\mathcal{L}$-formula ψ.

With a word model $\underline{w}$ as defined at the beginning of Section 4.2, an $\mathcal{M}$-$\mathcal{L}$-MSO-formula $\varphi(s_1, \ldots, s_k, S_1, \ldots, S_\ell)$ with free variables $s_1, \ldots, s_k$ and $S_1, \ldots, S_\ell$ additionally needs k positions $p_1, \ldots, p_k$ and ℓ sets of positions $P_1, \ldots, P_\ell$ such that we can say

$$(\underline{w}, p_1, \ldots, p_k, P_1, \ldots, P_\ell) \models \varphi(s_1, \ldots, s_k, S_1, \ldots, S_\ell)$$

if φ holds in $\underline{w}$ for the assignment $s_i = p_i$ and $S_j = P_j$ for $1 \leq i \leq k$ and $1 \leq j \leq \ell$. We also write

$$\underline{w} \models \varphi[p_1, \ldots, p_k, P_1, \ldots, P_\ell].$$

Given an $\mathcal{M}$-$\mathcal{L}$-MSO-sentence φ, we set $L(\varphi) = \{w \in M^n \mid \underline{w} \models \varphi\}$ as the language defined by φ. We call L $\mathcal{M}$-$\mathcal{L}$-MSO-definable if $L = L(\varphi)$ for some $\mathcal{M}$-$\mathcal{L}$-MSO-sentence φ.

Taking the alphabet frame $(\mathcal{M}, \mathcal{L})$ with $\mathcal{M} = (\mathbb{N}, +, 0)$ and $\mathcal{L} =$ FO-logic, we see, for example, that the language

$$L = \{w \in \mathbb{N}^+ \mid \text{the number of odd letters in } w \text{ is odd}\}$$

is $(\mathcal{M}, \mathcal{L})$-MSO-definable.

Now we are ready to establish the equivalences of $\mathcal{M}$-$\mathcal{L}$(-Büchi)-recognizability and $\mathcal{M}$-$\mathcal{L}$-MSO-definability for the case of finite respectively infinite words over $(\mathcal{M}, \mathcal{L})$, i.e., we show that a language $L \subseteq (M^n)^*$ is $\mathcal{M}$-$\mathcal{L}$-recognizable iff L is $\mathcal{M}$-$\mathcal{L}$-MSO-definable, resp. for the case of infinite words between $\mathcal{M}$-$\mathcal{L}$-Büchi-recognizable languages and $\mathcal{M}$-$\mathcal{L}$-MSO: $L \subseteq (M^n)^\omega$ is $\mathcal{M}$-$\mathcal{L}$-Büchi-recognizable iff L is $\mathcal{M}$-$\mathcal{L}$-MSO-definable. This equivalence is also effective if the MSO-theory of the underlying structure $\mathcal{M}$ is decidable.

We first treat the case of languages of finite words over $(\mathcal{M}, \mathcal{L})$ and show the equivalence by discussing both implications.

Lemma 4.15. *Let $\mathcal{B}$ be an $\mathcal{M}$-$\mathcal{L}$-automaton, then there exists an $\mathcal{M}$-$\mathcal{L}$-MSO sentence φ with $L(\mathcal{B}) = L(\varphi)$.*

PROOF. The proof works by an adaption of the classical case for finite alphabets shown by Büchi [Büc60], Elgot [Elg61], and Trakhtenbrot [Tra61] while additionally handling relations over infinite alphabets. Given an $\mathcal{M}$-$\mathcal{L}$-automaton $\mathcal{B} = (Q, M^n, q_1, \Delta, F)$ with $|Q| = m$, we have to construct an $\mathcal{M}$-$\mathcal{L}$-MSO sentence φ such that any word model $\underline{w}$ satisfies φ if and only if $\underline{w}$ is accepted by $\mathcal{B}$. Hence, the sentence φ has to express the existence of a successful run of $\mathcal{B}$ on $\underline{w}$. Therefore we introduce m sets $S_1, \dots, S_m$ representing the states of $\mathcal{B}$ and evaluating to true where a run of $\mathcal{B}$ on $\underline{w}$ passes through the respective state. We identify four properties that need to be expressed by φ in order to express the existence of a successful run of $\mathcal{B}$:

- At each point of time, the automaton is in only one state. Thus, the sets $S_1, \dots, S_m$ form a partition of $dom(w)$. This can be expressed by the $\mathcal{M}$-$\mathcal{L}$-formula

$$\varphi_{part}(S_1, \dots, S_m) = \forall s\big(\bigvee_{i=1}^{m} S_i(s) \wedge \bigwedge_{i \neq j, 1 \leq i,j \leq m} \neg(S_i(s) \wedge S_j(s))\big).$$

- The run of $\mathcal{B}$ starts in the initial state. This is easily expressed by

$$\varphi_{in} = \exists s(\neg \exists t(t < s) \wedge S_1(s)).$$

- The states that are assumed at consecutive positions s, t reflect the execution of a transition in Δ

$$\varphi_{trans} = \forall s \forall t\big(Suc(s,t) \rightarrow \bigvee_{(q_i, \psi, q_j) \in \Delta} (S_i(s) \wedge P_\psi(s) \wedge S_j(t))\big)$$

- The state assumed after the last letter of $\underline{w}$ is read is a final state.

$$\varphi_{fin} = \bigvee_{\exists q_j \in F : (q_i, \psi, q_j) \in \Delta} \big(\exists s(\neg \exists t(s < t) \wedge S_i(s) \wedge P_\psi(s))\big)$$

A conjunction over these formulas yields the desired sentence

$$\varphi = \exists S_1 \cdots \exists S_m (\varphi_{part}(S_1, \dots, S_m) \wedge \varphi_{in} \wedge \varphi_{trans} \wedge \varphi_{fin})$$

such that $\mathcal{B}$ accepts $\underline{w}$ iff $\underline{w} \models \varphi$. ∎

To simplify the implication from logic to automata, we introduce a variant of $\mathcal{M}$-$\mathcal{L}$-MSO-logic, in which first-order variables are replaced by singleton second-order variables; in complete analogy to the classical case of MSO_0-logic as defined in Chapter 2. We will modify $\mathcal{M}$-$\mathcal{L}$-MSO to the expressively equivalent formalism of $\mathcal{M}$-$\mathcal{L}$-MSO_0-formulas and then proceed by induction over MSO_0-formulas.

Definition 4.16. Given an alphabet frame $(\mathcal{M}, \mathcal{L})$, formulas of $\mathcal{M}$-$\mathcal{L}$-MSO$_0$-logic are built up from atomic formulas

$$S \subseteq T, S < T, \mathrm{Sing}(S), Suc(S,T), S \subseteq P_\psi \text{ for } \psi \in \Psi_n$$

as well as the connectives $\neg$, $\vee$, $\wedge$, and the set quantifiers $\exists$ and $\forall$ (where Ψ_n is the set of $\mathcal{L}$-formulas with n free variables).

Remark 4.17. Each $\mathcal{M}$-$\mathcal{L}$-MSO-formula can be transformed into an equivalent $\mathcal{M}$-$\mathcal{L}$-MSO$_0$-formula. ◂

PROOF. This translation from $\mathcal{M}$-$\mathcal{L}$-MSO-formulas to $\mathcal{M}$-$\mathcal{L}$-MSO$_0$-formulas can easily be shown via an induction over the structure of $\mathcal{M}$-$\mathcal{L}$-MSO-formulas.

We replace atomic formulas as follows:

- $T(s)$ by $\mathrm{Sing}(T) \wedge S \subseteq T$,
- $s < t$ by $\mathrm{Sing}(S) \wedge \mathrm{Sing}(T) \wedge S < T$,
- $Suc(s,t)$ by $\mathrm{Sing}(S) \wedge \mathrm{Sing}(T) \wedge Suc(S,T)$,
- $P_\psi(s)$ by $\mathrm{Sing}(S) \wedge S \subseteq P_\psi$,

and in the induction step, simply use the Boolean connectives as usual. For the translation of quantification, we will simply add the requirement that the variable in the scope of the quantifier, say s, is in fact a singleton set by adding the subformula $\mathrm{Sing}(S)$. ■

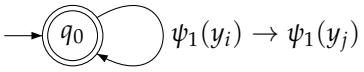
In order to translate an $\mathcal{M}$-$\mathcal{L}$-MSO$_0$-formula $\varphi(S_1, \ldots, S_m)$ into an equivalent $\mathcal{M}$-$\mathcal{L}$-automaton, as mentioned above we interpret the free variables $S_1, \ldots, S_m$ in a word model $\underline{w}$ and m sets $K_1, \ldots, K_m \subseteq dom(w)$. To code such models as a single word, we represent the sets $K_1, \ldots, K_m$ as bit vectors and thus let the automaton work with the above mentioned convolution on the alphabet $M^n \times \{0,1\}^m$.

Lemma 4.18. *Given an alphabet frame $(\mathcal{M}, \mathcal{L})$, let φ be an $\mathcal{M}$-$\mathcal{L}$-MSO-sentence. Then there exists an $\mathcal{M}$-$\mathcal{L}$-automaton $\mathcal{B}$ with $L(\varphi) = L(\mathcal{B})$.*

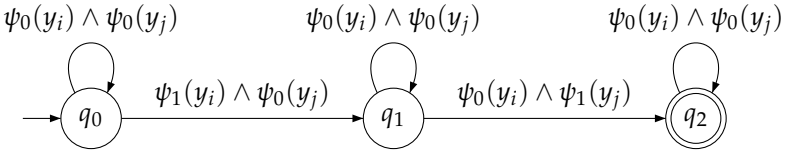
PROOF. We proceed by induction over the structure of $\mathcal{M}$-$\mathcal{L}$-MSO$_0$-formulas, i.e., for any $\mathcal{M}$-$\mathcal{L}$-MSO$_0$-formula $\varphi(S_1, \ldots, S_m)$, we have to specify an equivalent $\mathcal{M}$-$\mathcal{L}$-automaton $\mathcal{B}$ that recognizes the set of ω-words over $M^n \times \{0,1\}^m$ defined by this formula. Formally, every transition label $\psi(x_1, \ldots, x_n, y_1, \ldots, y_m)$ has free variables $x_1, \ldots, x_n, y_1, \ldots, y_m$. With a representation of a model $(\underline{w}, K_1, \ldots, K_m)$ as the input for the automaton, the upper n elements, which form the convolution $\langle w \rangle$ of w, are assigned to the free variables $x_1, \ldots, x_n$ while the lower m elements, which represent the bit vectors indicating membership in the sets $K_1, \ldots, K_m$, are assigned to the variables $y_1, \ldots, y_m$.

In the translation of atomic $\mathcal{M}$-$\mathcal{L}$-MSO$_0$-formulas into $\mathcal{M}$-$\mathcal{L}$-automata, we employ $\mathcal{M}$-$\mathcal{L}$-formulas $\psi_0(x)$ and $\psi_1(x)$ in the transitions of the $\mathcal{M}$-$\mathcal{L}$-automata, expressing that $x = 0$, respectively $x = 1$. This yields for the induction basis:

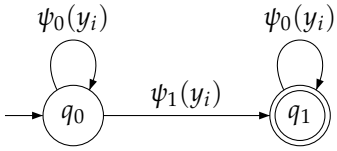
- $S_i \subseteq S_j$: The $\mathcal{M}$-$\mathcal{L}$-automaton checks that when the i-th bit component has entry 1, so does the j-th bit component.



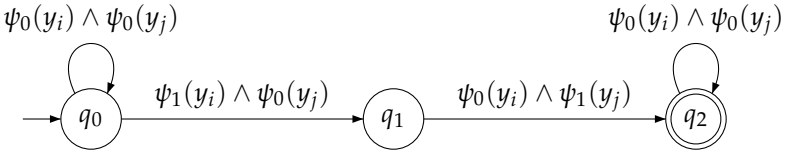
- $S_i < S_j$: The $\mathcal{M}$-$\mathcal{L}$-automaton checks that there is exactly one position such that the i-th and j-th bit component have entry 1, and that these occur in the right order.



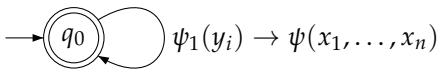
- Sing(S_i): The $\mathcal{M}$-$\mathcal{L}$-automaton ensures that there is precisely one position where the i-th bit component has entry 1.



- $Suc(S_i, S_j)$: Here, the $\mathcal{M}$-$\mathcal{L}$-automaton works in the same fashion as in the case $S_i < S_j$ while ensuring that the entries occur directly after one another.



- $S_i \subseteq P_\psi$ for $\psi \in \Psi_n$: The $\mathcal{M}$-$\mathcal{L}$-automaton checks that whenever the i-th bit component is 1, the upper n elements representing the current letter of the input word satisfy the $\mathcal{M}$-$\mathcal{L}$-formula ψ which defines the letter predicate P_ψ.



For the induction step, it suffices to consider the connectives $\vee$ and $\neg$, as well as the existential set quantifier $\exists$, since these are functionally complete (cf. e.g. [EFT07]). Here, we can exploit the closure properties of $\mathcal{M}$-$\mathcal{L}$-automata from Lemma 4.8, and employ the constructions for the union, complementation, and projection, respectively, which completes the proof. ■

Now we turn to ω-words and show that one can easily infer an equivalence between $\mathcal{M}$-$\mathcal{L}$-Büchi automata and $\mathcal{M}$-$\mathcal{L}$-MSO.

Remark 4.19. Let $\mathcal{B}$ be an $\mathcal{M}$-$\mathcal{L}$-Büchi automaton, then there exists an $\mathcal{M}$-$\mathcal{L}$-MSO sentence φ with $L(\mathcal{B}) = L(\varphi)$. ◂

PROOF. Again, the construction of an $\mathcal{M}$-$\mathcal{L}$-MSO formula describing a successful run of a given $\mathcal{M}$-$\mathcal{L}$-Büchi automaton $\mathcal{B}$ is a straightforward adaption of the well-known proof ([Tho97]).

We can use an analogous construction to the one we used for the case of finite words. Here, we only need to adapt the last formula expressing the acceptance condition to reflect a Büchi condition:

$$\varphi_{\mathit{fin}} = \forall s\big(\exists t(s < t \wedge \bigvee_{q_i \in F} S_i(t))\big).$$

Let us turn to the translation from $\mathcal{M}$-$\mathcal{L}$-MSO sentences to $\mathcal{M}$-$\mathcal{L}$-Büchi automata.

Proposition 4.20. *Let φ be an $\mathcal{M}$-$\mathcal{L}$-MSO sentence, then there exists an $\mathcal{M}$-$\mathcal{L}$-Büchi-automaton $\mathcal{B}$ with $L(\varphi) = L(\mathcal{B})$.*

PROOF. Again, we modify $\mathcal{M}$-$\mathcal{L}$-MSO to the expressively equivalent formalism of $\mathcal{M}$-$\mathcal{L}$-MSO_0-formulas and proceed by induction over MSO_0-formulas. This can be done in complete analogy to the case of finite words while invoking Lemmas 4.10 and 4.11 for the closure properties of $\mathcal{M}$-$\mathcal{L}$-Büchi automata. ■

As a summary of Remark 4.19 and Proposition 4.20, we can conclude the following.

Theorem 4.21. *A language $L \subseteq (M^n)^\omega$ with $n \geq 1$ of ω-words is $\mathcal{M}$-$\mathcal{L}$-MSO-definable iff it is $\mathcal{M}$-$\mathcal{L}$-Büchi-recognizable and the conversion of automata into formulas and vice versa is effective if the $\mathcal{L}$-theory of $\mathcal{M}$ is decidable.*

As a consequence of the $\mathcal{M}$-$\mathcal{L}$-Büchi theory, we obtain that satisfiability and equivalence of $\mathcal{M}$-$\mathcal{L}$-MSO-formulas over models from M^ω are decidable if the $\mathcal{L}$-theory of the structure $\mathcal{M}$ is decidable.

Theorem 4.22. *If the $\mathcal{L}$-theory of $\mathcal{M}$ is decidable, so are the satisfiability, equivalence, and inclusion problem over words resp. ω-words over $(\mathcal{M}, \mathcal{L})$.*

4.2.3 Strong Automata

In this section we extend the standard model of automata over infinite alphabets as introduced in the previous section. We address the essential weakness of that model, the inability to compare (or to modify) successive letters. In the present section we introduce the model of "strong automaton", in which, for instance, the aspect of checking for letters to be "clones" (i.e., to be identical to the previous letter) is included.

In a strong automaton (say over n-words) we use transitions that control the relation between two successive letters. Assume that $(\mathcal{M}, \mathcal{L})$ is an alphabet frame. A transition formula is now of the form $\psi_{p,q}(\overline{y}, \overline{x})$, and it can be applied to change from state p to state q via letter $\overline{a}$ from M^n if for the previous letter $\overline{a}_-$ of the input word we have

$$\mathcal{M} \models \psi[\overline{a}_-, \overline{a}]$$

We shall develop a theory of strong automata (and Büchi automata) in close analogy to the theory of standard automata as given in the previous section. However, two special features have to be mentioned: In order to preserve effectiveness results, we need

- to work with the MSO-theory of the alphabet structure $\mathcal{M}$ (rather than the $\mathcal{L}$-theory of $\mathcal{M}$ assumed to be decidable),
- to restrict to the case $n = 1$, i.e., to exclude the case of words over proper tuples of elements from M.

The first aspect reflects the fact that we need decidability of a theory of the alphabet structure which has a certain level of expressiveness.

Regarding the second aspect, we shall show that the nonemptiness problem for strong automata is undecidable over 2-words from $\mathbb{N}$, more precisely with the alphabet frame consisting of the successor structure $(\mathbb{N}, +1)$ and first-order logic.

In the subsequent sections we introduce strong automata (both over finite and infinite words). Then we show the mentioned undecidability result. In the remainder of the chapter we concentrate on the case of 1-words and show results for strong automata in analogy to standard $\mathcal{M}$-$\mathcal{L}$-automata.

4.2.3.1 Definitions

Definition 4.23. Let $\mathcal{M}$ be a structure with domain M. A strong automaton over $(\mathcal{M}, \mathcal{L})$ handling *finite words* of n-tuples of M-elements is of the form

$$\mathcal{B} = (Q, M^n, q_0, \Delta, F)$$

where

- Q is a finite set of states,
- M^n is the input alphabet,
- $q_0 \in Q$ is the initial state,
- $\Delta \subseteq Q \times (\Psi_n \cup \Psi_{2n}) \times Q$ is the finite transition relation, where Ψ_n resp. Ψ_{2n} are the sets of $\mathcal{L}$-formulas with n resp. $2n$ free variables,
- and $F \subseteq Q$ is the set of accepting states.

For strong automata over $(\mathcal{M}, \mathcal{L})$, the underlying structure $\mathcal{M}$ contains a clone predicate C. The format is the same as for the standard automata over $(\mathcal{M}, \mathcal{L})$ mentioned above, except for the transition relation $\Delta \subseteq Q \times (\Psi_n \cup \Psi_{2n}) \times Q$. In order to capture the clone predicate, we define for each state pair (p, q) the possible transitions via a formula $\psi_{pq}(y_1, \ldots, y_n, x_1, \ldots, x_n)$ – or, in the special case of an initial transition, via a formula $\psi_{q_0 q}(x_1, \ldots, x_n)$. Starting with the latter case, the strong automaton can proceed from q_0 to q with input letter $\overline{a}_1$, if $\mathcal{M} \models \psi_{q_0 q}[\overline{a}_1(1), \ldots, \overline{a}_1(n)]$. For a transition of the first case, in which a previous input letter exists and is $\overline{a}_{i-1}$, the strong automaton can move from p to q if $\mathcal{M} \models \psi_{pq}[\overline{a}_{i-1}(1), \ldots, \overline{a}_{i-1}(n), \overline{a}_i(1), \ldots, \overline{a}_i(n)]$. Again, a run ρ is successful if $\rho(m) \in F$.

Let us note that the clone predicate C_i (applied to the i-th component of a letter, and satisfied when this component is equal to the i-th component of the preceding letter) is definable in terms of strong automata: We just use the transition formula

$$\psi_{pq}(y_1, \ldots, y_n, x_1, \ldots, x_n) := x_i = y_i.$$

Let us list some examples of languages recognized by strong automata:

- over the alphabet frame $(\mathcal{M}, \mathcal{L})$ with $\mathcal{M} = (\mathbb{N}, +, <, 0)$, $\mathcal{L} =$ FO-logic:
  - the language $\{0\,1\,2 \cdots i \mid i \geq 0\}$,
  - the set of words containing a pair of equal successive letters,
- and taking $\mathcal{M} = (\mathbb{N}, +, \cdot, 0, 1)$
  - the set of words $\{p_0 p_1 \cdots p_i \mid i \geq 0\}$ where p_i is the i-th prime.

Similarly, we define strong Büchi automata over $(\mathcal{M}, \mathcal{L})$.

Definition 4.24. For an alphabet frame $(\mathcal{M}, \mathcal{L})$, a strong Büchi automaton over $(\mathcal{M}, \mathcal{L})$ handling n-tuples of M-elements is of the form

$$\mathcal{B} = (Q, M^n, q_0, \Delta, F)$$

where Q, M^n, q_0, F are as above and

- $\Delta \subseteq Q \times (\Psi_n \cup \Psi_{2n}) \times Q$ is the finite transition relation, where Ψ_n resp. Ψ_{2n} are the sets of $\mathcal{L}$-formulas with n resp. $2n$ free variables.

We define for each state pair (p, q) the possible transitions via a formula

$$\psi_{pq}(y_1, \ldots, y_n, x_1, \ldots, x_n) \in \Psi_{2n},$$

and allow formulas of Ψ_n only for the special case of an initial transition by a formula $\psi_{q_0q}(x_1, \ldots, x_n)$. If $\alpha = \langle \alpha_1, \ldots, \alpha_n \rangle$ is an ω-word over M^n, a *run* of $\mathcal{B}$ on α is an infinite sequence of states $\rho = \rho(0)\rho(1)\ldots$ with $\rho(0) = q_0$ such that there exists an $\mathcal{M}$-$\mathcal{L}$-formula $\psi_{q_0q}(x_1, \ldots, x_n)$ and a transition $(\rho(0), \psi_{q_0q}, \rho(1))$ satisfying $\mathcal{M} \models \psi_{q_0q}[\alpha_1(0), \ldots, \alpha_n(0)]$ and for every $i > 0$ there exists an $\mathcal{M}$-$\mathcal{L}$-formula $\psi_{pq}(y_1, \ldots, y_n, x_1, \ldots, x_n)$ and a transition $(\rho(i), \psi_{pq}, \rho(i+1))$ satisfying

$$\mathcal{M} \models \psi_{pq}[\alpha_1(i-1), \ldots, \alpha_n(i-1), \alpha_1(i), \ldots, \alpha_n(i)]$$

A run ρ of $\mathcal{B}$ on α is *successful* if there exist infinitely many i such that $\rho(i) \in F$. We say that $\mathcal{B}$ *accepts* α if there exists a successful run of $\mathcal{B}$ on α. We denote by $L(\mathcal{B})$ the set of ω-words over M^n accepted by $\mathcal{B}$.

In the next section we show an undecidability result that prohibits an effective theory for strong automata (or Büchi automata) when the dimension of the considered words is at least 2. This motivates a restriction to dimension 1 that is pursued in the subsequent section. It will turn out that we can develop results as for standard $(\mathcal{M}, \mathcal{L})$-automata when $\mathcal{L} = \text{MSO}$ and assuming that the MSO-theory of $\mathcal{M}$ is decidable.

4.2.3.2 An Undecidability Result

Theorem 4.25. *Let $(\mathcal{M}, \mathcal{L})$ be the alphabet frame with $\mathcal{M} = (\mathbb{N}, +1, 0)$ and $\mathcal{L} =$ first-order logic. The nonemptiness problem for strong automata over $(\mathcal{M}, \mathcal{L})$ is undecidable.*

PROOF. We use a reduction from the halting problem for 2-register machines.

Such a machine R is given by a finite sequence

$$1\ \text{instr}_1; \ldots; k-1\ \text{instr}_{k-1}; k\ \text{stop}$$

where each instruction instr_j is of the form

- $\mathrm{Inc}(X_1)$, $\mathrm{Inc}(X_2)$ (increment the value of X_1, respectively X_2 by 1), or
- $\mathrm{Dec}(X_1)$, $\mathrm{Dec}(X_2)$ (similarly for decrement by 1, with the convention that a decrement of 0 is 0), or
- If $X_i = 0$ goto ℓ_1 else to ℓ_2 (where $i = 1, 2$ and $1 \leq \ell_1, \ell_2 \leq k$, with the natural interpretation).

An R-configuration is a triple (ℓ, m, n), indicating that the ℓ-th instruction is to be executed and the values of X_1, X_2 are m, n, respectively. A terminating R-computation (for R as above) is a sequence $(\ell_0, m_0, n_0), \ldots, (\ell_r, m_r, n_r)$ of M-configurations where in each step the update is done according to the instructions in R and the last instruction is the stop-instruction (formally: $\ell_r = k$). The termination problem for 2-counter machines asks to decide, for any given 2-counter machine R, whether there exists a terminating R-computation that starts with $(1, 0, 0)$ (abbreviated as $R : (1, 0, 0) \to \text{stop}$). It is well-known that the termination problem for 2-counter machines is undecidable ([Min67]). So we use the undecidability of the following problem:

Given a 2-register machine R with stop line k, decide whether
$$R : (1, 0, 0) \to (k, 0, 0).$$

Our task is to exhibit a computable transformation $R \mapsto \mathcal{A}_R$ where $\mathcal{A}_R$ is a strong automaton, and two states of it, called p_1 and p_k, such that

$(*)$ $\quad R : (1, 0, 0) \to (k, 0, 0)$ iff $\mathcal{A}_R$ has a run from p_1 to p_k over $\mathbb{N}^2$

For this we let $\mathcal{A}_R$ build up the R-computation starting from configuration $(1, 0, 0)$. In fact, $\mathcal{A}_R$ works as follows: From the initial state q_0, $\mathcal{A}_R$ can move only via input letter $(0, 0)$ to state p_1.

From state p_i the automaton will be able to move via the input (m_1, m_2) to p_j, where (k_1, k_2) is the previous letter, iff R proceeds from configuration (i, k_1, k_2) to (j, m_1, m_2) in one step. If this is guaranteed, the claim $(*)$ is proved.

The implementation thereof is not hard when we supply first-order formulas $\psi_{p_i, p_j}(y_1, y_2, x_1, x_2)$ for the transitions. For example, if the i-th R-instruction is $\mathrm{Inc}(X_1)$ then we define

$$\psi_{p_i p_{i+1}}(y_1, y_2, x_1, x_2) := \quad y_1 = x_1 + 1$$

and if it is

$$\text{If } X_1 = 0 \text{ goto } j \text{ else to } j'$$

we introduce

$$\psi_{p_i p_j}(y_1, y_2, x_1, x_2) := \quad x_1 = 0$$

and

$$\psi_{p_i p_{j'}}(y_1, y_2, x_1, x_2) := \quad x_1 > 0;$$

similarly in the other cases.

So $\mathcal{A}_R$ has state set $\{q_0, p_1, \ldots, p_k\}$, initial state q_0, transitions as introduced above, and p_k as single finite state. ■

As a consequence of the undecidability result of the preceding section, we enter here a study of strong automata (and Büchi automata) over words of dimension 1.

4.2.3.3 Closure properties and Nonemptiness Problem for Dimension $n = 1$

We now want to capture basic closure properties of the class of languages recognized by strong automata.

To obtain effectiveness for the closure properties to be shown, we have to assume a stronger statement than that the alphabet frame $(\mathcal{M}, \mathcal{L})$ is effective. This assumption allowed us to check availability of a transition $p \xrightarrow{\psi(\overline{x})} q$ by a satisfiability check of ψ in $\mathcal{M}$. For strong automata we have to cover sequences of transitions. In order to achieve this, we let $\mathcal{M}$ be arbitrary but fix $\mathcal{L}$ to be MSO-logic, and we assume that the MSO-theory of $\mathcal{M}$ is decidable.

First we treat the nonemptiness problem.

The nonemptiness problem For the case of one-dimensional strong automata we now show that the nonemptiness problem is decidable over the alphabet frame $(\mathcal{M}, \mathcal{L})$, provided $\mathcal{L} =$ MSO-logic and the MSO-theory of $\mathcal{M}$ is decidable. First we treat the case of strong automata over finite words, then the case of infinite words.

Theorem 4.26. *For an alphabet frame $(\mathcal{M}, \text{MSO})$ where the MSO-theory of $\mathcal{M}$ is decidable, we have the following results.*

(a) The nonemptiness problem for strong automata over $(\mathcal{M}, \text{MSO})$ handling finite words is decidable.

(b) The nonemptiness problem for strong Büchi automata over $(\mathcal{M}, \text{MSO})$ is decidable.

PROOF. Proof of (a)

Assume the MSO-theory of $\mathcal{M}$ is decidable.

Let $\mathcal{A} = (Q, M, q_0, \Delta, F)$ be a strong automaton. The aim is to exhibit an MSO-formula $\varphi_{\mathcal{A}}$ over the structure $\mathcal{M}$ such that

$$\mathcal{M} \models \varphi_{\mathcal{A}} \text{ iff } \mathcal{A} \text{ has a (finite) successful run over } M.$$

For this purpose, we introduce a binary relation E over M:

$$E(a, a') :\Leftrightarrow \text{iff there is a transition formula}$$
$$\psi_{p,q}(y, x) \text{ of } \mathcal{A} \text{ such that } \mathcal{M} \models \psi_{p,q}[a, a'].$$

In this case we write $\mathcal{A} : a, p \to a', q$

By E^+ we denote the irreflexive transitive closure of E. Note that we can express $E^+(a, a')$ in MSO over M by an MSO-formula formalizing the following:

"for all Y containing a
and closed under taking E-successors we have $a' \in Y$".

With the desired sentence $\varphi_{\mathcal{A}}$ we want to express the following:

There is an element a_1 (as first input of a successful run) and further elements $a_2, \ldots, a_k$ as continuation of the input that satisfy $E(a_i, a_{i+1})$ for $0 \leq i < k$, allowing a run from q_0 to a final state.

The association of states to the input is implemented by a partition of the set $S := \{a_1, \ldots, a_k\}$ into sets S_q for $q \in Q$:

$$S_q = \{a' \in S \mid \exists a \exists p \text{ such that } \mathcal{A} : a, p \to a', q\}.$$

In order to guarantee that the set S induces a run starting with initial input letter a_1, we require:

- S only contains elements a with $E^+(a_1, a)$

  (This fixes the tree of those elements that are reachable from a_1 via E)

- For each element $s \in S$ there is precisely one element $t \in S$ such that $E(s, t)$

  (This makes S a path of M-elements rather than a tree)

Then we have to say that the partition into the sets S_q is compatible with the transition relation of $\mathcal{A}$:

"for all $a, a' \in S$ with $E(a, a')$ we have
$\bigvee_{p,q} a \in S_p \wedge a' \in S_q \wedge \mathcal{A} : a, p \to a', q$ (i.e., $\mathcal{M} \models \psi_{p,q}[a, a']$)".

The final condition says that the element a_1 is obtained by an initial transition $\mathcal{A} : q_0 \to a_1, q_1$, so $a_1 \in S_{q_1}$, and that, for some $q \in F$, an element of S_q occurs in S. This finishes the construction of $\varphi_{\mathcal{A}}$.

Proof of (b)

The proof is identical to part (a) except that the very last condition is replaced by a formalization of the Büchi condition

$$\bigvee_{q \in F} \forall s \in S(\exists t \in S_q(E^+(s, t))).$$

■

Now we address closure properties for strong automata.

Lemma 4.27. *The class of strong $\mathcal{M}$-$\mathcal{L}$-recognizable languages (of finite words) is closed under union, projection, and complementation.*

PROOF. The closure properties of strong $\mathcal{M}$-$\mathcal{L}$-recognizable languages (of finite words) are shown by slight adaptions of the classical case (where the alphabet is finite). Here, we concentrate on pointing out the adaptions rather than the actual constructions. For example, an automaton for the projection from M^n to M^{n-1} can easily be obtained by replacing the "label" $\psi(x_1, \ldots, x_n)$ of a transition by $\exists x_n \psi(x_1, \ldots, x_n)$. For the complementation, we follow the strategy of a determinization via a powerset construction and then simply swapping the sets F and $Q \setminus F$ (as outlined in [Bès08]). The idea is as follows: Given an $\mathcal{M}$-$\mathcal{L}$-automaton $\mathcal{B}$ (on finite words), $\mathcal{B}$ does not necessarily provide a run (accepting or not accepting) for every possible input letter in M^n, i.e., there might be a letter that does not satisfy any of the formulas of the transitions. For the construction of the complement automaton, one modifies the set of formulas for the transitions such that each input word leads to a complete run, and additionally, one prepares for determinism: Let $\psi_1, \ldots, \psi_m$ be the formulas which occur in the transitions of $\mathcal{B}$. For each subset $J \subseteq \{1, \ldots, k\}$, introduce the formula $\chi_J := \bigwedge_{i \in J} \psi_i \wedge \bigwedge_{i \notin J} \neg \psi_i$. Note that for $J \neq J'$, there is no symbol $\bar{a} \in M^n$ with $\mathcal{M} \models \chi_J \wedge \chi_{J'}[\bar{a}]$, and for each $\bar{a}$, there is a set J such that $\mathcal{M} \models \chi_J[\bar{a}]$. Then we construct $\mathcal{B}'$ by replacing each transition $(p, \psi_i, q) \in \Delta$ by (p, Ψ_i, q) with $\Psi_i = \bigvee_{J \ni i} \chi_J$. Then $L(\mathcal{B}') = L(\mathcal{B})$, and one can continue with the usual powerset construction. ∎

Lemma 4.28. *If the* MSO*-theory of $\mathcal{M}$ is decidable, the class of ω-languages $L \subseteq M^\omega$ recognized by strong $\mathcal{M}$-*MSO*-Büchi automata over M is effectively closed under the Boolean operations and definable projections $p : M \to M$.*

PROOF. This claim is shown in precise analogy to the case of standard $\mathcal{M}$-$\mathcal{L}$-Büchi automata (and we skip here the repetition of proofs), except for the closure under complement. Here we describe the necessary modifications.

The approach is the same as for the standard case, i.e., via Büchi's original method involving finite colorings and Ramsey's theorem. However, the coloring of a segment of an ω-word over the alphabet M^n, i.e., the transition profile, is defined differently. Given a strong Büchi automaton $\mathcal{A}$, the "strong transition profile" of the segment $\alpha[i, j]$ of an ω-word α refers also to the last previous letter $\alpha(i-1)$ if $i > 0$. This extra context information is needed in order to capture the clone predicate on the n components of α, and we define the transition profile of a segment relative to this context information within α. So an appropriate notation for a strong transition profile is $tp_\alpha([i, j])$ rather than $tp(u)$. Such profiles, however, are of the same type as the previously defined profiles (namely, presented as two sets of pairs of states). The transition profile of a segment $\alpha[i, j]$ is fixed from the state pairs (p, q) that allow a run of the automaton from p to q (respectively, a

run from p to q via a final state), where in the first move the letter $\alpha(i-1)$ is used (This condition is dropped for the case $i = 0$).

There is, of course, a definite conceptual difference to the usual coloring of segments in terms of standard transition profiles: There, one may concatenate any sequence of segments (for given transition profiles) to obtain a new composed segment whose transition profile is induced by the given ones. In the new setting, the composition of segments u and v only works when the clone information on the last letter of u agrees with the first letter of v. However, this does not affect the argument in Büchi's complementation proof: Here we only need that for any *given* α one can obtain a sequence $i_0 < i_1 < \ldots$ such that all segments $\alpha[i_j, i_{j+1} - 1]$ share the same transition profile, and that for such a sequence, the transition profiles of $\alpha[0, i_0 - 1]$ and of $\alpha[i_0, i_1 - 1]$ determine α either to be accepted or not to be accepted by the Büchi automaton.

Also the sets $U_{\vartheta_0} \cdot U_{\vartheta}^{\omega}$ can be used as before when defined properly: Such a set is not obtained by freely concatenating a segment $u \in U_{\vartheta_0}$ and a sequence of segments from U_{ϑ}; rather, it is the set

$$U_{\vartheta_0} \cdot U_{\vartheta}^{\omega} = \{\alpha \mid \exists i_0, i_1, \ldots (0 < i_0 < i_1 < \ldots \wedge tp_{\alpha}[0, i_0 - 1] = \vartheta_0 \\ \wedge tp_{\alpha}[i_j, i_{j+1} - 1] = \vartheta \text{ for } j = 0, 1, \ldots)\}$$

The effective presentation of the complement of $L(\mathcal{A})$ is now completed as in the preceding subsection for $\mathcal{M}$-$\mathcal{L}$-Büchi automata. ■

As a consequence of Lemma 4.27 and Lemma 4.28 we obtain the following result.

Proposition 4.29. *If the* MSO*-theory of $\mathcal{M}$ is decidable, the inclusion problem and the equivalence problem for strong $\mathcal{M}$-*MSO*-Büchi recognizable languages are decidable.*

4.2.4 Comparing Distant Letters

As seen before in Section 4.2.3.2, strong automata that allow us to handle more than one dimension of an alphabet frame $(\mathcal{M}, \mathcal{L})$, i.e., allowing the alphabet to be M^n for $n > 1$, already have an undecidable nonemptiness problem. Thus, in order to have a theory of strong automata with nice closure properties, we needed to restrict ourselves to dimension 1.

A characteristic feature of strong automata is the definability of the clone predicate C. It holds for position t of a word model if the letter there is equal to the previous letter. We showed decidability of the nonemptiness problem when the MSO-theory of $\mathcal{M}$ is decidable.

We will now show, that as soon as we try to extend the clone predicate in another manner, we again lose decidability. The extension of our model now

allows to compare not only subsequent/successive letters of a word, but also more distant ones, i.e., items which are not direct successors. For this we recall the "clone-on-distance relation" $\sim$. Undecidability then is reached even if we restrict ourselves pretty thoroughly, which facilitates the case considered. We will employ a "liberal version" of the clone-on-distance relation. We work in a logical framework rather than with automata, since the undecidability only involved FO-logic over the word models (rather than MSO-logic, which would motivate to work with automata).

Thus, we let the alphabet structure $\mathcal{M}$ be the structure over the natural numbers with addition and constant 0: $\mathcal{M} = (\mathbb{N}, +, 0)$. As the aforementioned logic $\mathcal{L}$ which describes properties of elements of $\mathcal{M}$, we take first-order logic, thus $\mathcal{L} = \text{FO}$. As "global" structure, we already reach undecidability by considering finite words (instead of infinite trees) over the alphabet structure $\mathcal{M}$, and restricting ourselves to again first-order logic as the logic which captures properties of the positions in a word over $\mathcal{M}$. But we will extend the word signature, which usually contains the predicates *Suc* and $<$ by a "neighbor-on-distance" predicate $\sim^1$ over positions.

Definition 4.30. We define the "neighbor-on-distance" relation $\sim^1$ over words from $\mathbb{N}^+$ as follows:

Given $w = a_0 \cdots a_{\ell-1} \in \mathbb{N}^+$ we have $s \sim^1 t$ if $|a_s - a_t| = 1$ for $0 \leq s, t \leq \ell - 1$ and $\ell - 1 \in \mathbb{N}$.

Thus, we allow a variation of the "classical" clone predicate which is able to compare/touch the values stored at positions s and t allowing values that differ by 1. Therefore, we will refer to the employed letter position logic as FO-logic over $(\mathcal{M}, \mathcal{L})$-words with $\sim^1$, written FO-DIST$_1$, which is FO-logic over $(\mathcal{M}, \mathcal{L})$, which on top of $\mathcal{L}$-predicates is allowed to use the neighbor-on-distance predicate $\sim^1$. We will show that this logic has an undecidable satisfiability problem:

Theorem 4.31. *Satisfiability of* FO-DIST$_1$*-formulas for* $(\mathbb{N}, +, 0)$-FO *properties in finite words over* $\mathbb{N}$ *is undecidable.*

For the proof we use a reduction from Post's correspondence problem PCP [Pos46], which is well-known to be undecidable (the classic proof is provided in [Pos46]; for a modern proof we refer the reader to [Sip97]). In an effort to ease notation, we will be considering PCP over the alphabet $A = \{a, b\}$ only.

Starting with the definitions of the aforementioned relations and word model, we recall PCP and then reduce our satisfiability problem to it.

Definition 4.32. Given the alphabet frame $(\mathcal{M}, \mathcal{L})$ with $\mathcal{M} = (\mathbb{N}, +, 0)$ and $\mathcal{L} = \text{FO}$, we define the extended word model for a word $w = a_0 \cdots a_{\ell-1} \in M^+$ as

follows:

$$\underline{w} = (\{0, \ldots, \ell - 1\}, Suc, <, (P_\psi)_{\psi \in \Psi}, \sim^1)$$

where Suc, $<$, and $(P_\psi)_{\psi \in \Psi}$ are defined as usual, and $\sim^1$ is defined as above.

Definition 4.33. We recall Post's Correspondence Problem PCP over a finite alphabet A with $|A| = 2$:

Input: Two finite lists of words $\overline{u} = (u_1, \ldots, u_k)$ and $\overline{v} = (v_1, \ldots, v_k)$ with $u_i, v_i \in A^+$ for $1 \leq i \leq k$ and $k \in \mathbb{N}$.
Problem: Is there a sequence of indices $1 \leq i_1 \leq \ldots \leq i_r \leq k$ such that $u_{i_1} \cdots u_{i_r} = v_{i_1} \cdots v_{i_r}$?

We will call $u_{i_1} \cdots u_{i_r}$ the corresponding solution word for the PCP-instance $(\overline{u}, \overline{v})$.

It is well-known that this decision problem is undecidable.

Definition 4.34. Given an alphabet frame $(\mathcal{M}, \mathcal{L})$, we consider the satisfiability problem FO-DIST-SAT for FO-DIST$_1$-logic over finite $(\mathcal{M}, \mathcal{L})$ words:

Input: An FO-DIST$_1$-sentence φ over finite $(\mathcal{M}, \mathcal{L})$-words.
Problem: Is there a finite $\mathcal{M}$-word w such that $\underline{w} \models \varphi$?

Now, we are ready to reduce PCP to FO-DIST-SAT.

Lemma 4.35. *Given a* PCP *instance* $\big((u_1, \ldots, u_k), (v_1, \ldots, v_k)\big)$ *over* $A = \{a, b\}$, *one can construct an* FO-DIST$_1$*-sentence* $\varphi_{(\overline{u},\overline{v})}$ *over finite* $((\mathbb{N}, +, 0), \text{FO})$ *words such that*

$(\overline{u}, \overline{v})$ *has a solution* $\Leftrightarrow$ *there is a finite* $((\mathbb{N}, +, 0), \text{FO})$*-word* w *such that* $w \models \varphi_{(\overline{u},\overline{v})}$.

PROOF. In order to realize the reduction PCP $\leq$ FO-DIST-SAT, we need a transformation of a PCP-instance $(\overline{u}, \overline{v})$ over alphabet $A = \{a, b\}$ into a FO-DIST$_1$-sentence $\varphi_{(\overline{u},\overline{v})}$ over finite $((\mathbb{N}, +, 0), \text{FO})$-words such that

$$(\overline{u}, \overline{v}) \text{ has a solution } \Leftrightarrow \text{ there is a finite } \mathcal{M}\text{-word } w \text{ such that } w \models \varphi_{(\overline{u},\overline{v})}$$

For this, we encode the existence of a solution $i_1 < \cdots < i_r$ in two steps: first, we encode a solution word $u_{i_1} \cdots u_{i_r} = v_{i_1} \cdots v_{i_r}$ by a word model over $\tilde{A} = A \cup \underline{A}$ with $\underline{A} = \{\underline{a} \mid a \in A\}$ (in our case, this yields $\tilde{A} = \{a, b, \underline{a}, \underline{b}\}$). Then we again code this word by numbers such that in conclusion we receive a word model over $\mathbb{N}$.

In the first step, we interleave a solution word securing the following form:

$$u_{i_1} \underline{v}_{i_1} \cdots u_{i_r} \underline{v}_{i_r} \in \tilde{A}^*$$

where $\underline{v} = \underline{v}_{i_1} \cdots \underline{v}_{i_r} \in \underline{A}^*$ is the copy of $v \in A^*$ with marked (underlined) letters.

With this format, we use the second encoding to enhance the information provided at each position. We will encode two pieces of information into unique

numbers in $\mathbb{N}$ for each position: first, which letter of $\tilde{A}$ is encoded (this will be called the "basic code" of the given letter), and second, we provide information to uniquely identify the letter of A (resp. $\underline{A}$) given at the current position in order to match it to its corresponding occurrence of $\underline{A}$ (resp. A). Such a corresponding pair of letters will be encoded within a unique interval of size $8 = 2 \cdot |\tilde{A}|$ and be the only numbers in the resulting word over $\mathbb{N}$ within this interval.

Thus, a letter c of the resulting word w over $\mathbb{N}^+$ has the format $c = m \cdot 8 + n$ where n is the basic code of the letter of $\tilde{A}$ to be encoded, while m determines the unique interval for this letter and its counterpart. With $\tilde{A} = \{a, b, \underline{a}, \underline{b}\}$ we identify letter a with basic code 0, letter $\underline{a}$ with basic code 1, letter b with basic code 2, and letter $\underline{b}$ with basic code 3; consequently, each letter a in u_{i_j} will be coded by a number $\equiv 0 \mod 8$ and so on. Note that each corresponding pair of letters is coded by two numbers of distance 1.

Remark 4.36. If we work with modulus $|\tilde{A}|$ (in this case 4) rather than $2 \cdot |\tilde{A}|$ (in this case 8), there would be numbers of distance 1 that are of the form $m \cdot 4 + 0$ and $(m-1) \cdot 4 + 3$ which would confuse the reference. ◀

Following these preparations, we have that PCP instance $(\overline{u}, \overline{v})$ has a solution iff the FO-DIST$_1$-sentence $\varphi_{(\overline{u},\overline{v})}$ expressing the following properties has a $((\mathbb{N}, +, 0), \text{FO})$-word model:

1. each number $c \in w$ is a valid code
2. two successive maximal blocks B, C (coding unmarked, resp. marked letters) of numbers code one of the pairs (u_i, v_i)
3. there exists precisely one "corresponding" letter for each position
4. for all pairs of corresponding letters identified above: the same letters are coded, one marked, the other nonmarked.
5. the right ordering among the corresponding pairs of letters needs to be ensured by excluding the case that the positions of one pair of corresponding numbers encompasses the positions of another pair (i.e., for pairs $s_1 < t_1$ and $s_2 < t_2$, we do not have $s_1 < s_2 < t_2 < t_1$)

We can express all of these requirements by the following FO-DIST$_1$-sentences over positions in the $((\mathbb{N}, +, 0), \text{FO})$-word w:

1. Being a valid code means that for each number c in w we have $c \equiv p \mod 8$ for $0 \leq p \leq 3$:

$$\varphi_1 = \forall s(P_{\psi_0(x)}(s) \vee P_{\psi_1(x)}(s) \vee P_{\psi_2(x)}(s) \vee P_{\psi_3(x)}(s))$$

2. With s indicating the beginning of an unmarked block B, for a pair (u_i, v_i) let k_i denote the length of u_i and κ_i denote the length of v_i; let p_{i_j} be the basic code of a word (u_{i_j}) and q_{i_j} be the basic code of a word (v_{i_j}). Within the two successive maximal blocks, B consists of numbers $\equiv 0 \mod 8$ and $\equiv 2 \mod 8$, and C consists of numbers $\equiv 1 \mod 8$ and $\equiv 3 \mod 8$. Thus we obtain the following formula for the identification of two successive maximal blocks B, C with one of the pairs (u_i, v_i):

$$\begin{aligned}\varphi_2 = \forall s\Big(& \big(\neg\exists t(t < s) \vee \exists t(s = Suc(t) \wedge P_{\psi_{\text{marked}}(x)}(t))\big) \\ & \to \bigvee_{1\le i\le k} \Big(\bigwedge_{0\le j<k_i} (P_{\psi_{p_{i_j}}}(Suc^j(s))) \\ & \wedge \bigwedge_{0\le j<\kappa_i} (P_{\psi_{q_{i_j}}}(Suc^{k_i+j}(s)))\Big) \\ & \wedge \big(\neg\exists t(t = Suc^{k_i+\kappa_i}(s)) \vee \neg P_{\psi_{\text{marked}}(x)}(Suc^{k_i+\kappa_i}(s))\big)\Big)\end{aligned}$$

3. We can identify precisely one "corresponding" letter for each position with the neighbor-on-distance relation:

$$\varphi_3 = \forall s\exists t\big(s \sim^1 t \wedge s \neq t \wedge \neg\exists t'(t' \neq t \wedge t' \neq s \wedge s \sim^1 t')\big)$$

4. To ensure the right coding (marked and nonmarked) of the corresponding letters, we simply invoke the matching predicates defined by formulas of the alphabet logic:

$$\begin{aligned}\varphi_4 = \forall s\forall t\Big(& s \sim^1 t \to \big(P_{\psi_0(x)}(s) \wedge P_{\psi_1(x)}(t)\big) \\ & \vee \big(P_{\psi_1(x)}(s) \wedge P_{\psi_0(x)}(t)\big) \vee \big(P_{\psi_2(x)}(s) \wedge P_{\psi_3(x)}(t)\big) \\ & \vee \big(P_{\psi_3(x)}(s) \wedge P_{\psi_2(x)}(t)\big)\Big)\end{aligned}$$

5. Ensuring the right ordering among corresponding pairs of letters is straightforward when invoking the neighbor-on-distance relation:

$$\begin{aligned}\varphi_5 = \forall s_1\forall s_2\forall t_1\forall t_2\big(& s_1 < t_1 \wedge s_2 < t_2 \wedge s_1 < s_2 \\ & \wedge s_1 \sim^1 t_1 \wedge s_2 \sim^1 t_2 \to t_1 < t_2\big)\end{aligned}$$

The predicates used above are expressed in the alphabet logic FO over $\mathcal{M} = (\mathbb{N}, +, 0)$ as follows:

- expressing that $x \equiv p \mod 8$:

$$\psi_p(x) = \exists y(x = y + y + y + y + y + y + y + y + p) \text{ for } 0 \le p \le 3$$

- expressing that x is a marked letter:

$$\psi_{\text{marked}}(x) = \psi_1(x) \vee \psi_3(x)$$

If the word model fulfills this formula it has the form $u_{i_1}\underline{v}_{i_1}\cdots u_{i_r}\underline{v}_{i_r} \in \tilde{A}^*$, and thus, $u_{i_1}\cdots u_{i_r} = v_{i_1}\cdots v_{i_r}$ holds. This concludes the proof of Theorem 4.31. ■

Example 4.37. Consider the following PCP-instance over alphabet $A = \{a, b\}$ with $(\overline{u}, \overline{v}) = ((u_1, u_2u_3), (v_1, v_2, v_3))$:

	1	2	3
$\overline{u}$	b	bab	ba
$\overline{v}$	bab	aba	b

A corresponding solution word is

$$u_3u_2u_3u_1 = ba \cdot bab \cdot ba \cdot b = bababbab = b \cdot aba \cdot b \cdot bab = v_3v_2v_3v_1$$

According to the construction detailed above, the coding format of the solution word is

$$ba\underline{b}bab\underline{aba}bab\underline{b}b\underline{bab} \in \tilde{A}^*$$

With $\tilde{A} = \{a, b, \underline{a}, \underline{b}\}$, we have the basic codes 0 for letter a, 1 for letter $\underline{a}$, 2 for letter b, and 3 for letter $\underline{b}$. Therefore, with interval size 8, this leaves us to calculate the code c of a letter with basic code n by $c = m \cdot 8 + n$, where m determines the unique interval for this occurrence and its counterpart:

word	b	a	$\underline{b}$	b	a	b	$\underline{a}$	$\underline{b}$	$\underline{a}$	b	a	$\underline{b}$	b	$\underline{b}$	$\underline{a}$	$\underline{b}$
(n, m)	$(2,1)$	$(0,2)$	$(3,1)$	$(2,3)$	$(0,4)$	$(2,5)$	$(1,2)$	$(3,3)$	$(1,4)$	$(2,6)$	$(0,7)$	$(3,5)$	$(2,8)$	$(3,6)$	$(1,7)$	$(3,8)$
code	10	16	11	26	32	42	17	27	33	50	56	43	66	51	57	67

Thus, the resulting word over $\mathbb{N}^+$ is

$$w = 10 \quad 16 \quad 11 \quad 26 \quad 32 \quad 42 \quad 17 \quad 27 \quad 33 \quad 50 \quad 56 \quad 43 \quad 66 \quad 51 \quad 57 \quad 67,$$

which codes the given solution word for PCP-instance $(\overline{u}, \overline{v})$ and is a model of the FO-DIST$_1$-sentence $\varphi_{(\overline{u},\overline{v})}$. ◀

4.3 Tree Models

In this final section, we turn to tree structures obtained by two different kinds of "tree iteration" of a given relational structure $\mathcal{M}$. We introduce these tree iterations, denoted $\mathcal{M}^\#$ and $\mathcal{M}^*$, in Section 4.3.1. In the subsequent sections we investigate chain logic over $\mathcal{M}^\#$ and $\mathcal{M}^*$ with the additional equal-level predicate.

4.3.1 Iterating Relational Structures

We consider relational structures with finite signature. Such a structure is presented in the format $\mathcal{M} = (M, R_1, \ldots, R_k)$ where R_i is of arity $r_i > 0$. We focus on structures called "admissible": In this case there are two designated elements (usually called 0 and 1), represented by two singleton predicates P_0, P_1 that belong to the tuple $(R_1, \ldots, R_k)$. Then we can view bit sequences as special sequences over $\mathcal{M}$.

We introduce two tree models built from a relational structure $\mathcal{M}$. The first is the weak tree iteration

$$\mathcal{M}^{\#} = (M^*, \preceq, Suc, R_1^*, \ldots, R_k^*)$$

where $u \preceq v :\Leftrightarrow u$ is a prefix of v, Suc is the successor relation containing all pairs (u, ua) with $u \in M^*, a \in M$, and for every R_i, say of arity ℓ, we have $R_i^*(v_1, \ldots, v_\ell)$ iff there exists $z \in M^*, a_1, \ldots a_\ell \in M$ such that $v_j = za_j$ for $1 \leq j \leq \ell$ and $R_i(a_1, \ldots, a_\ell)$. (In [Bès08] a variant of this definition is used, namely that there exist $z_1, \ldots, z_\ell \in M^*$ of same length and $a_1, \ldots, a_\ell \in M$ such $v_j = z_j a_j$ with $R_i(a_1, \ldots, a_\ell)$.)

The strong tree iteration of $\mathcal{M}$ is the structure

$$\mathcal{M}^* = (M^*, \preceq, Suc, R_1^*, \ldots, R_k^*, C)$$

where everything is as above for $\mathcal{M}^{\#}$ and $C = \{u\ a\ a \mid u \in M^*, a \in M\}$. The expansions of $\mathcal{M}^{\#}$, $\mathcal{M}^*$ by the equal-level relation E (with $E(u, v)$ iff $|u| = |v|$) are denoted $\mathcal{M}_E^{\#}$, $\mathcal{M}_E^*$, respectively.

If $\mathcal{M}$ is finite, we assume that each individual letter of M is definable. The usual approach is to introduce a constant in the signature of $\mathcal{M}$ for each element of M. In the present paper we stick to relational structures and use a singleton predicate R_a for each element $a \in M$. So the binary alphabet $\{0, 1\}$ is coded by the structure $\mathcal{M}_2 = (\{0, 1\}, R_0, R_1)$ with $R_0 = \{0\}$, $R_1 = \{1\}$. In the case of finite structures $\mathcal{M}$ there is no essential difference between $\mathcal{M}^{\#}$ and $\mathcal{M}^*$, since the clone predicate C becomes definable in $\mathcal{M}^{\#}$ by the equivalence

$$C(v) \leftrightarrow \bigvee_{a \in M} (\exists u (R_a^*(u) \wedge S(u, v) \wedge R_a^*(v)).$$

As a consequence we note that for finite $\mathcal{M}$, the structures $\mathcal{M}^{\#}$ and $\mathcal{M}^*$ coincide (in the sense that the clone predicate becomes definable).

We recall chain logic over the tree structures $\mathcal{M}^{\#}$ and $\mathcal{M}^*$ built from $\mathcal{M}$. A path (through the tree domain M^*) is a maximal set linearly ordered by $\preceq$; it may be identified with an ω-word in M^ω, obtained as the common extension of all the words $u \in M^*$ forming the path. A chain is a subset of a path, and we can view

a singleton set in M^* as a chain. We call chain logic the fragment of MSO-logic in which set quantification is restricted to chains.

As indicated in previous chapters, it is convenient to eliminate first-order variables and quantifiers in terms of (singleton) chain quantifiers. This simplifies the setting since only one kind $S_1, S_2, \ldots$ of variables remains, ranging over chains. In order to simulate first-order logic, the signature of tree models has to be adapted. As atomic formulas one uses

- $\mathrm{Sing}(S)$ for "S is a singleton"
- $S_i \subseteq S_j$ with its standard meaning,
- $\mathrm{Succ}(S_i, S_j)$ for "S_i is a singleton $\{s_i\}$, S_j is a singleton $\{s_j\}$, with $Suc(s_i, s_j)$; similarly for $S_i \preceq S_j$.

The resulting formalism is called chain_0 logic; it has the same expressive power as chain logic.

As shown by Shelah and Stupp [She75, Stu75], respectively Muchnik and Walukiewicz (see the announcement in [Sem84] and the proof in [Wal02]), the MSO-theory of $\mathcal{M}^\#$ and the MSO-theory of $\mathcal{M}^*$ are decidable if the MSO-theory of $\mathcal{M}$ is. In the present thesis we show the decidability of the chain logic theory of structures $\mathcal{M}_E^\#$, obtained by adjoining the equal-level relation E to $\mathcal{M}^\#$, under mild assumptions on the structure $\mathcal{M}$. Our results extend work of Kuske and Lohrey [KL06] on structures $\mathcal{M}^\#$ and of Bès [Bès08] on structures $\mathcal{M}_E^\#$. Furthermore, we show – in contrast to the Muchnik-Walukiewicz result for MSO-logic – that a transfer of this decidability result to tree structures $\mathcal{M}_E^*$ is not possible.

Bès shows the decidability of the chain logic theory of $\mathcal{M}_E^\#$ if the first-order theory of $\mathcal{M}$ is decidable. Here we refine his result: We refer to any logic $\mathcal{L}$ such that the $\mathcal{L}$-theory of $\mathcal{M}$ is decidable, and we consider an *extension* of the chain theory of $\mathcal{M}^\#$ in which further quantifications are allowed, namely quantifiers of $\mathcal{L}$ restricted to the set of siblings of any element v. (Thus one allows quantifiers over elements x that are Suc-successors of any given element v.) We call the corresponding theory the *chain logic theory of $\mathcal{M}_E^\#$ with $\mathcal{L}$ on siblings*. We show that this theory is decidable if the $\mathcal{L}$-theory of $\mathcal{M}$ is.

In our framework two logics play together: The logic $\mathcal{L}$ allows to express relations between $\mathcal{M}$-elements as they appear as sons of some given node of the tree, and chain logic is used to speak about (sets of) tree elements arranged along paths. Referring to the standard graphical representation of trees, $\mathcal{L}$ captures the horizontal dimension and chain logic the vertical dimension. On the level of signatures, the predicate E of the tree signature refers to the horizontal while the successor and the prefix relation refer to the vertical aspect; finally, the signature of $\mathcal{M}$ enters in the horizontal dimension, restricted to the children of a tree node.

Definition 4.38. Given an alphabet frame $(\mathcal{M}, \mathcal{L})$, we define *chain logic with $\mathcal{L}$ on siblings* over tree iterations of $\mathcal{M}$ as the usual chain logic expanded by predicates P_ψ, where ψ is an $\mathcal{L}$-formula, and for a tree iteration τ we define

$$\tau \models P_\psi(x_1, \ldots, x_n) \Leftrightarrow \tau \models \exists x(\bigwedge_{1 \leq i \leq n} Suc(x, x_i) \wedge \mathcal{M} \models \psi_{\restriction x}(x_1, \ldots, x_n))$$

where $\psi_{\restriction x}$ indicates that the $\mathcal{L}$-formula ψ is relativized to the successors of node x in τ.

For an admissible alphabet M (containing two identifiable elements 0,1) we encode a chain c as a pair $\hat{c} := (\alpha, \beta) \in (M^\omega)^2$ where

- α encodes the path of which c is a subset. As c can be finite, we set α to be the path $a_0 \ldots a_r 000 \ldots$ where a_r is the last c-element of which c is a subset; it can be interpreted as a sequence of "directions". Note that for each element w in c it holds that w is a prefix of α.
- β codes membership in c along the path α, i.e., $\beta(i) = 1$ iff $\alpha[0, i] \in c$.

So if $c = \emptyset$, α is the path 0^ω through the tree M^* and β also is the sequence that is constant 0.

The technical treatment below is simplified when we let an n-tuple $(\alpha_1, \ldots, \alpha_n)$ of ω-words over M be perceived as a single ω-word over M^n, the *convolution* of $(\alpha_1, \ldots, \alpha_n)$:

$$\langle \alpha_1, \ldots, \alpha_n \rangle := \begin{bmatrix} \alpha_1(0) \\ \vdots \\ \alpha_n(0) \end{bmatrix} \begin{bmatrix} \alpha_1(1) \\ \vdots \\ \alpha_n(1) \end{bmatrix} \cdots \quad \in (M^n)^\omega$$

Similarly, we define the *convolution of a relation $R \subseteq (M^\omega)^n$ of ω-words* to be the ω-language

$$L_R := \{\langle \alpha_1, \ldots, \alpha_n \rangle \mid (\alpha_1, \ldots, \alpha_n) \in R\}.$$

So the n-tuples of M-elements just considered will be used as letters of ω-words and input letters of Büchi automata. Transitions of automata will be specified in a logic $\mathcal{L}$ by means of $\mathcal{L}$-formulas $\psi(x_1, \ldots, x_n)$. Each of these formulas defines a unary predicate $\psi^\mathcal{M}$ over M^n:

$$\psi^\mathcal{M} = \{(a_1, \ldots, a_n) \in M^n \mid \mathcal{M} \models \psi[a_1, \ldots, a_n]\}$$

In general we consider ω-models over M^n for a signature that is given by a finite set Ψ of $\mathcal{L}$-formulas: Given a tuple $(\alpha_1, \ldots, \alpha_n)$ of words over an alphabet M and a finite set Ψ of $\mathcal{L}$-formulas $\psi_1, \ldots, \psi_k$ with n free variables each, we define the structure

$$\underline{\langle \alpha_1, \ldots, \alpha_n \rangle} = (\mathbb{N}, 0, <, Suc, (P_\psi)_{\psi \in \Psi})$$

with the usual interpretations of $0, <, Suc$ (the latter for the successor relation), and the letter predicates $P_\psi = \{i \in \mathbb{N} \mid (\alpha_1(i), \ldots, \alpha_n(i)) \in \psi^{\mathcal{M}}\}$. Thus, P_ψ collects all letter positions of $\langle \alpha_1, \ldots, \alpha_n \rangle$ which carry a letter from M^n that shares the property described by ψ.

For these ω-models over $\mathcal{M}$, equipped with predicates P_ψ defined in $\mathcal{L}$, we shall use a generalized form of MSO-logic, where – as usual in ω-language theory – the first-order quantifiers range over $\mathbb{N}$ and the monadic second-order quantifiers over sequences of letters (here from M). The system will be called $\mathcal{M}$-$\mathcal{L}$-MSO.

For an $\mathcal{M}$-$\mathcal{L}$-MSO-sentence φ, where the predicates P_ψ are introduced via $\mathcal{L}$-formulas $\psi(x_1, \ldots, x_n)$ with n free variables, we set

$$L(\varphi) = \{\langle \alpha_1, \ldots, \alpha_n \rangle \in (M^n)^\omega \mid \underline{\langle \alpha_1, \ldots, \alpha_n \rangle} \models \varphi\}$$

as the ω-language defined by φ. We say a relation $R \subseteq (M^\omega)^n$ is $\mathcal{M}$-$\mathcal{L}$-MSO *definable* if there is a $\mathcal{M}$-$\mathcal{L}$-MSO sentence φ with $L_R = L(\varphi)$.

Later on, it will be convenient to refer to the component entries of an ω-word $\langle \alpha_1, \ldots, \alpha_n \rangle$ in a more readable way than via an index $i \in \{1, \ldots, n\}$. So, when a sequence variable T is used for the i-th component α_i, we shall write $T(s)$ to indicate the element $\alpha_i(s)$ for $s \in \mathbb{N}$.

Analogous definitions can be given for the case of finite words over M^n.

4.3.2 Weak Tree Iterations

In this section, we want to show that for the weak tree iteration $\mathcal{M}_E^\#$ with equal-level relation, the chain theory of $\mathcal{M}_E^\#$ is decidable with $\mathcal{L}$ on siblings.

With the preparations of Section 4.2.2, we will establish a reduction from chain logic formulas over tree models to $\mathcal{M}$-$\mathcal{L}$-MSO over ω-sequences (and then to Büchi automata).

To avoid heavy notation, we employ chain$_0$ logic as introduced in Chapter 2, and provide the following construction. Recall that for a chain c in $\mathcal{M}_E^\#$, the object $\hat{c}$ is a pair of sequences over M coding the path underlying the chain c, respectively the membership of nodes of this path in c.

Lemma 4.39. *For any chain$_0$-formula $\varphi(S_1, \ldots, S_n)$ over the weak tree iteration $\mathcal{M}_E^\# = (M^*, Suc, \preceq, R_1^*, \ldots, R_k^*, E)$ with $\mathcal{L}$ on siblings, one can construct an $\mathcal{M}$-$\mathcal{L}$-MSO-formula $\varphi'(T_1^P, S_1^C, \ldots, T_n^P, S_n^C)$ interpreted in ω-words over M^{2n} such that for all chains $c_1, \ldots, c_n$ we have:*

$$\mathcal{M}_E^\# \models \varphi[c_1, \ldots, c_n]$$
$$\text{if and only if } \underline{\langle \hat{c}_1, \ldots, \hat{c}_n \rangle} \models \varphi'(T_1^P, S_1^C, \ldots, T_n^P, S_n^C).$$

PROOF. We proceed by induction over the structure of chain$_0$-formulas with $\mathcal{L}$ on siblings over $\mathcal{M}_E^\#$.

For the induction basis we have to consider the atomic formulas, namely of the form $\mathrm{Sing}(S)$, $S_i \subseteq S_j$, $S_i \preceq S_j$, $R_i^*(S_1, \ldots, S_k)$, $E(S_i, S_j)$, and also the $\mathcal{L}$-formulas $\psi(x_{i_1}, \ldots, x_{i_\ell})$.

As a first example, we present the translation into $\mathcal{M}$-$\mathcal{L}$-MSO-formulas for the formula $\varphi(S) = \mathrm{Sing}(S)$: Given the encoding $\hat{c} = (\alpha, \beta)$ of a chain c, the formula $\varphi'_{\mathrm{Sing}}(S)$ has to express that β indicates membership in c exactly once. Thus, we obtain $\varphi'_{\mathrm{Sing}}(T^P, S^C) = \exists s\big(S^C(s) \wedge \forall t(t \neq s \to \neg S^C(s))\big)$.

For the case of an $\mathcal{L}$-formula $\psi(x_{i_1}, \ldots, x_{i_\ell})$, we capture $x_{i_1}, \ldots, x_{i_\ell}$ by corresponding singletons $X_{i_1}, \ldots, X_{i_\ell}$, and these in turn by pairs $(T^P_{i_1}, S^C_{i_1}), \ldots, (T^P_{i_\ell}, S^C_{i_\ell})$ consisting of a path $T^P_{i_j} \in M^\omega$ and a singleton set indicator $S^C_{i_j} \subseteq \{0,1\}^\omega$ each. We have to define a corresponding predicate $P_\psi \subseteq ((M \times \{0,1\})^n)^\omega$ by an $\mathcal{M}$-$\mathcal{L}$-MSO-formula that expresses in terms of the $T^P_{i_j}$, $S^C_{i_j}$ that there is a common *Suc*-predecessor z of the elements x_{i_j} and that the tuple $x_{i_1}, \ldots, x_{i_\ell}$ satisfies ψ. In intuitive notation, we have

$$\varphi'_{P_\psi}(T^P_1, S^C_1, \ldots, T^P_n, S^C_n) = \bigwedge_{j=1}^{\ell} \text{“}(T^P_{i_j}, S^C_{i_j}) \text{ is singleton containing } x_{i_j}\text{”} \\ \wedge \exists z \bigwedge_{j=1}^{\ell} \text{“}\mathit{Suc}(z, x_{i_j})\text{”} \wedge \psi(x_{i_1}, \ldots, x_{i_\ell})$$

In some more detail:

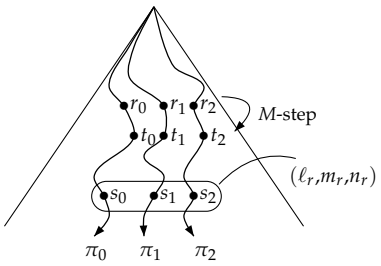
$$\bigwedge_{j=1}^{\ell} \varphi'_{\mathrm{Sing}}(T^P_{i_j}, S^C_{i_j}) \wedge \exists x_{i_1} \ldots \exists x_{i_\ell} \exists s\big(S^C_{i_j}(s) \wedge T^P_{i_j}(s) = x_{i_j} \\ \wedge \bigwedge_{j' \neq j} \forall t < s\,(T^P_{i_j}(t) = T^P_{i_{j'}}(t)) \wedge \psi(x_{i_1}, \ldots, x_{i_\ell})\big)$$

The induction step then is straightforward, as $\mathcal{M}$-$\mathcal{L}$-MSO is closed under the Boolean operations and projection. ∎

Thus, we obtain a reduction of the chain$_0$-theory with $\mathcal{L}$ on siblings of $\mathcal{M}_E^\#$ to the $\mathcal{M}$-$\mathcal{L}$-MSO theory, which with Theorem 4.21 is decidable if the $\mathcal{L}$-theory of $\mathcal{M}$ is decidable. This leaves us to conclude this section with the following theorem:

Theorem 4.40. *If the $\mathcal{L}$-theory of $\mathcal{M}$ is decidable, the chain-theory of $\mathcal{M}_E^\#$ with $\mathcal{L}$ on siblings is decidable.*

Figure 4.1 Coding a computation of a 2-register machine by three paths in a tree.



4.3.3 Strong Tree Iterations

We turn to the model-checking problem over structures $\mathcal{M}_E^*$. We show undecidability even for the first-order theory of $\mathcal{M}_E^*$ when $\mathcal{M}$ is the structure $\mathcal{S} := (\mathbb{N}, \mathrm{Succ})$ (where Succ is successor).

Theorem 4.41. *The first-order theory of $\mathcal{S}_E^*$ with* FO *on siblings is undecidable.*

PROOF. For any 2-register machine $\mathcal{R}$ we construct a first-order formula $\varphi_{\mathcal{R}}$ with FO on siblings such that $\mathcal{R} : (1,0,0) \to$ stop iff $\mathcal{S}_E^* \models \varphi_{\mathcal{R}}$.

The idea is to code a computation $(\ell_0, m_0, n_0), \ldots, (\ell_r, m_r, n_r)$ by three finite paths of same length, one for each of the three components (see Figure 4.1). Each of these paths (namely $\pi_0 = (\ell_0, \ldots, \ell_r), \pi_1 = (m_0, \ldots, m_r), \pi_2 = (n_0, \ldots, n_r)$) is determined by its last point in the tree structure $\mathcal{S}_E^*$, i.e., by a triple v_0, v_1, v_2 of $\mathcal{S}_E^*$-elements.

We use a formula which expresses

$$\exists s_0 \exists s_1 \exists s_2 (E(s_0, s_1) \wedge E(s_1, s_2) \\ \wedge [s_0, s_1, s_2 \text{ code a terminating computation of } \mathcal{R}]).$$

In order to obtain a formalization of the condition in squared brackets, we have to express

1. the initial condition that π_0 starts with the son 1 of the root and π_1, π_2 with the son 0 of the root,

2. the progress condition that for each $r_0 \prec s_0$ (giving an instruction number), the corresponding $\mathcal{R}$-instruction is executed, which involves the vertex r_0 and the vertices $r_1 \prec s_1, r_2 \prec s_2$ on the same level as r_0 and their respective successors t_0, t_1, t_2 on π_0, π_1, π_2, respectively (see again Figure 4.1),

3. the termination condition that s_0 is the number k.

Accordingly, we can formalize the condition in squared brackets by a conjunction of three formulas φ_1, φ_2, φ_3 in the free variables r_0, r_1, r_2, making use of the (definable) tree successor relation *Suc*.

- The formula φ_1 expresses (in first-order logic with FO on siblings) for the root r of the tree model and those three *Suc*-successors r_0, r_1, r_2, where $r_0 \preceq s_0$, $r_1 \preceq s_1$, $r_2 \preceq s_2$, that r_0 is the number 1 and r_1, r_2 are the number 0 (of the model $\mathcal{S} = (\mathbb{N}, \mathrm{Succ})$).

- The formula φ_2 is of the form:

  > "for all $r_0 \prec s_0$, $r_1 \prec s_1$, $r_2 \prec s_2$ with $E(r_0, r_1)$ and $E(r_0, r_2)$, there are tree-successors t_0, t_1, t_2 (i.e., with $\mathit{Suc}(r_0, t_0)$, $\mathit{Suc}(r_1, t_1)$, $\mathit{Suc}(r_2, t_2)$ with $t_0 \preceq s_0$, $t_1 \preceq s_1$, $t_2 \preceq s_2$) that represent the correct update of the configuration (r_0, r_1, r_2)."

  The condition on update is expressed by a disjunction over all program instructions; we present, as an example, the disjunction member for the statement "3 $\mathrm{Inc}(X_2)$":

  $$r_0 \text{ is number 3 in } (\mathbb{N}, \mathrm{Succ}) \rightarrow t_0 \text{ is number 4 in } (\mathbb{N}, \mathrm{Succ})$$
  $$\wedge\, t_1 \text{ is the clone of } r_1 \wedge t_2 \text{ is the Succ-successor of the clone of } r_2.$$

  It is easy to formalize this in first-order logic with FO on siblings, similarly for the Dec-instructions and the jump instructions.

- The formula φ_3 expresses the third condition and is clearly formalizable in first-order logic with FO on siblings. ■

Let us turn to the second undecidability result. We shall confine ourselves to the simplest setting, where the structure $\mathcal{M}$ is just $(\{0,1\}, \{0\}, \{1\})$, i.e., $\mathcal{M}_E^{\#}$ and $\mathcal{M}_E^{*}$ are *both* the binary tree with equal-level relation (see also [Tho09]).

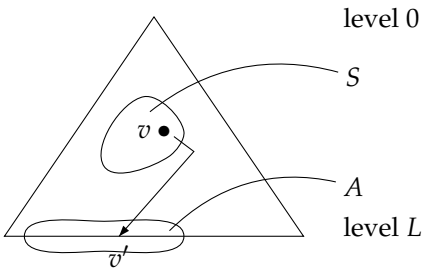
We refer to "chain logic with MSO on tree levels". In this logic we are allowed to use MSO-subformulas that are restricted to a tree level L.

Theorem 4.42. *The chain theory of the binary tree with equal-level relation and* MSO *on tree levels is undecidable.*

PROOF. We use an idea of [PT93] that allows to code a tuple of finite sets of the binary tree up to (and excluding) level L by a tuple of subsets of level L itself. In other words, we code a subset S of tree nodes before level L by an "antichain" A which is a subset of the level L (see Figure 4.2).

We simply map a vertex v (before level L) to the unique vertex $v' \in L$ which belongs to $v10^*$ (i.e., belongs to the leftmost path from the right successor of v;

Figure 4.2 Coding an element of a set S by an element of an antichain A.



see again Figure 4.2). The map $v \mapsto v'$ is injective and definable in chain logic (even in FO-logic), given the level L. Moreover, it is easy to see that the relations of being left or right successor in the tree are translated to FO-definable relations over the level L under consideration.

Using this coding, an existential quantifier over finite sets in the binary tree is captured by an existential quantifier over subsets of an appropriate level of the tree (namely, of a level that is beyond all maximal elements of the finite set under consideration).

Thus, the weak MSO-theory of the binary tree with E is interpretable in the FO-theory of the binary tree $(\{0,1\}^*, Suc_0, Suc_1, \preceq, E)$ with E and with MSO restricted to levels.

Since the weak MSO-theory of the binary tree with E is undecidable (see e.g. [Tho90]), we obtain the claim. ■

Intuitively, this result says that even over the binary tree the two dimensions of quantification (over chains and over tree levels) are very powerful and exceed the expressive power of weak MSO-logic.

Chapter 5

Conclusion

Looking Back

In this thesis we presented results that extend the automata-theoretic methods for studying definability of properties of words and trees.

In the first main part, we analyzed chain logic as a natural fragment of MSO-logic which arises in many applications. We suggested an automata-theoretic description of tree properties that are definable in chain logic – using the model of ranked alternating tree automaton with a condition on the admissible transitions that match the idea of building up paths or chains.

In the second part, we offered theories of automata and logic over words that are formed from an infinite alphabet. The main emphasis of our study was a clean separation between two logics involved: one logic, usually denoted $\mathcal{L}$, for describing properties of letters, and one for describing (finite or infinite) sequences of letters (i.e., words or ω-words). We showed how to generalize from the case of finite alphabets (which is handled by Boolean logic), with a focus on decidability properties and different mechanisms to establish connections between successive letters. It turned out that for showing algorithmic properties there are rather tight limitations.

Further Research

Let us mention three major issues left open in the present work:

- The model of ranked alternating automaton was considered here only over binary trees. An extension to k-ary trees for larger k does not involve principal problems but requires more technical effort. However, the extension to infinitely branching trees seems non-trivial and raises a challenge which is not addressed in the present thesis.

- Another way to merge aspects that are treated in the two parts of the thesis is to study tree structures (rather than words) over infinite alphabets. Most of the "positive" results proved here for words will have a counterpart in the domain of trees. To explore this in a systematic way is left here for future work.

- Finally, the relations we considered in the second part of the thesis for establishing connections between different letters of a word (over an infinite alphabet) are only first natural examples. A much more comprehensive theory should be developed in which the concept of "clone" and of "neighbor-on-distance" appear as special cases.

Bibliography

[BDM+06] Mikołaj Bojanczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-Variable Logic on Words with Data. In *Proceedings of the 21th IEEE Symposium on Logic in Computer Science, LICS 2006*, pages 7–16. IEEE Computer Society, 2006. 42

[Bès08] Alexis Bès. An Application of the Feferman-Vaught Theorem to Automata and Logics for Words over an Infinite Alphabet. *Computing Research Repository*, abs/0801.2498, 2008. 3, 42, 48, 50, 68, 75, 76

[BF85] Jon Barwise and Solomon Feferman. *Model-Theoretic Logics*. Perspectives in Mathematical Logic. Springer, 1985. 12, 45

[BFG05] Michael Benedikt, Wenfei Fan, and Floris Geerts. XPath Satisfiability in the Presence of DTDs. In Chen Li, editor, *Proceedings of the 24th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS 2005*, pages 25–36. ACM, 2005. 42

[Büc60] J. Richard Büchi. Weak Second-Order Arithmetic and Finite Automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960. 1, 13, 15, 56, 58

[Büc62] J. Richard Büchi. On a Decision Method in Restricted Second Order Arithmetic. In Ernest Nagel, Patrick Suppes, and Alfred Tarski, editors, *Proceedings of the 1960 International Congress on Logic, Methodology and Philosophy of Science, LMPS 1960*, pages 1–11. Stanford University Press, June 1962. 1, 11, 16, 54, 56

[CDG+07] Hubert Comon, Max Dauchet, Rémi Gilleron, Christof Löding, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. TATA: Tree Automata Techniques and Applications. Available on: `http://tata.gforge.inria.fr/`, 2007. release October, 12th 2007. 14

[CW98] Bruno Courcelle and Igor Walukiewicz. Monadic Second-Order Logic, Graph Coverings and Unfoldings of Transition Systems. *Annals of Pure and Applied Logic*, 92(1):35–62, 1998. 4, 43

[Don70] John Doner. Tree Acceptors and some of their Applications. *Journal of Computer and System Science*, 4(5):406–451, 1970. 1, 14

[EFT07] Heinz-Dieter Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Einführung in die mathematische Logik*. Spektrum Akademischer Verlag, Heidelberg, 5 edition, 2007. 10, 12, 45, 61

[Elg61] Calvin C. Elgot. Decision Problems of Finite Automata Design and Related Arithmetics. *Transactions of the American Mathematical Society*, 98:21–52, 1961. 1, 13, 56, 58

[GRST96] Dora Giammarresi, Antonio Restivo, Sebastian Seibert, and Wolfgang Thomas. Monadic Second-Order Logic over Rectangular Pictures and Recognizability by Tiling Systems. *Information and Computation*, 125(1):32–45, 1996. 3, 22

[GS84] Ferenc Gécseg and Magnus Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984. 14

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002. 16, 17, 34

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979. 7, 12

[KL06] Dietrich Kuske and Markus Lohrey. Monadic Chain Logic over Iterations and Applications to Pushdown Systems. In *Proceedings of the 21st Symposium on Logic in Computer Science, 2006*, pages 91–100. IEEE Computer Society, 2006. 76

[MH84] Satoru Miyano and Takeshi Hayashi. Alternating Finite Automata on omega-Words. *Theoretical Computer Science*, 32:321–330, 1984. 16

[Min67] Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967. 65

[MS84] David E. Muller and Paul E. Schupp. Alternating Automata on Infinite Objects, Determinacy and Rabin's Theorem. In Maurice Nivat and Dominique Perrin, editors, *Automata on Infinite Words, Ecole de Printemps d'Informatique Théorique, 1984*, volume 192 of *Lecture Notes in Computer Science*, pages 100–107. Springer, 1984. 16, 25

[MS87] David E. Muller and Paul E. Schupp. Alternating Automata on Infinite Trees. *Theoretical Computer Science*, 54:267–276, 1987. 27

[MSS86] David E. Muller, Ahmed Saoudi, and Paul E. Schupp. Alternating Automata. The Weak Monadic Theory of the Tree, and its Complexity. In Laurent Kott, editor, *Proceedings of the 13th International Colloquium on Automata, Languages and Programming, ICALP 1986*, volume 226 of *Lecture Notes in Computer Science*, pages 275–283. Springer, 1986. 39

[Mul63] David E. Muller. Infinite Sequences and Finite Machines. In *Proceedings of the 4th Annual Symposium on Switching Circuit Theory and Logical Design, FOCS 1963*, pages 3–16. IEEE Computer Society, 1963. 15

[Pos46] Emil L. Post. A Variant of a Recursively Unsolvable Problem. *Journal of Symbolic Logic*, 12(2):255–56, 1946. 70

[PT93] Andreas Potthoff and Wolfgang Thomas. Regular Tree Languages without Unary Symbols are Star-Free. In *Proceedings of the 9th International Symposium on Fundamentals of Computation Theory, FCT '93*, volume 710 of *Lecture Notes in Computer Science*, pages 396–405. Springer, 1993. 81

[Rab69] Michael O. Rabin. Decidability of Second-Order Theories and Automata on Infinite Trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969. 1, 17, 21, 22, 41

[Rab70] Michael O. Rabin. Weakly Definable Relations and Special Automata. In Yehoshua Bar-Hillel, editor, *Proceedings of the Symposium on Mathematical Logic and Foundations of Set Theory, SMLFST 1970*, pages 1–23. North-Holland, 1970. 16, 39

[Rab72] Michael O. Rabin. *Automata on Infinite Objects and Church's Problem*. Regional Conference Series in Mathematics. Conference Board of the Mathematical Sciences, 1972. 17

[Ram30] Frank P. Ramsey. On a Problem in Formal Logic. *Proceedings of the London Mathematical Society*, 30:264–286, 1930. 55

[Sem84] Alexei L. Semenov. Decidability of Monadic Theories. In Michal Chytil and Václav Koubek, editors, *Proceedings of the 11th International Symposium on Mathematical Foundations of Computer Science, MFCS 1984*, volume 176 of *Lecture Notes in Computer Science*, pages 162–175. Springer, 1984. 4, 5, 43, 76

[She75] Saharon Shelah. The monadic theory of order. *Annals of Mathematics*, 102:379–419, 1975. 4, 5, 76

[Sip97] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997. 70

[Stu75] Jonathan Stupp. The Lattice-Model is Recursive in the Original Model. Technical report, Institute of Mathematics, The Hebrew University, Jerusalem, January 1975. 4, 5, 76

[STW11] Alex Spelten, Wolfgang Thomas, and Sarah Winter. Trees over Infinite Structures and Path Logics with Synchronization. In Fang Yu and Chao Wang, editors, *Proceedings of the 13th International Workshop on Verification of Infinite-State Systems, INFINITY 2011*, volume 73 of *EPTCS*, pages 20–34, 2011. 43

[SVW87] A. Prasad Sistla, Moshe Y. Vardi, and Pierre Wolper. The Complementation Problem for Büchi Automata with Applications to Temporal Logic. *Theoretical Computer Science*, 49:217–237, 1987. 54

[Tho84] Wolfgang Thomas. Logical Aspects in the Study of Tree Languages. In Bruno Courcelle, editor, *Proceedings of the 9th Colloquium on Trees in Algebra and Programming, CAAP 1984*, pages 31–50. Cambridge University Press, 1984. 2, 39

[Tho87] Wolfgang Thomas. On Chain Logic, Path Logic, and First-Order Logic over Infinite Trees. In *Proceedings of the 2nd Annual Symposium on Logic in Computer Science, LICS 1987*, pages 245–256. IEEE Computer Society, 1987. 2, 5, 43

[Tho90] Wolfgang Thomas. Automata on Infinite Objects. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, volume B, pages 133–192. Elsevier Science Publishers, 1990. 13, 16, 17, 19, 41, 82

[Tho91] Wolfgang Thomas. On Logics, Tilings, and Automata. In Javier L. Albert, Burkhard Monien, and Mario Rodríguez-Artalejo, editors, *Proceedings of the 18th International Colloquium on Automata, Languages and Programming, ICALP 1991*, volume 510 of *Lecture Notes in Computer Science*, pages 441–454. Springer, 1991. 22

[Tho92] Wolfgang Thomas. Infinite Trees and Automation-Definable Relations over omega-Words. *Theoretical Computer Science*, 103(1):143–159, 1992. 3

[Tho97] Wolfgang Thomas. Languages, Automata, and Logic. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 389–455. Springer, 1997. 56, 61

[Tho09] Wolfgang Thomas. Path Logics with Synchronization. In Kamal Lodaya, Madhavan Mukund, and R. Ramanujam, editors, *Perspectives in Concurrency Theory*, IARCS-Universities, pages 469–481. Universities Press, 2009. 81

[Tra61] Boris A. Trakhtenbrot. Finite Automata and the Logic of Monadic Predicates. *Doklady Akademii Nauk SSSR*, 140:326–329, 1961. In Russian. 1, 13, 56, 58

[TW68] James W. Thatcher and Jesse B. Wright. Generalized Finite Automata Theory with an Application to a Decision Problem of Second-Order Logic. *Mathematical Systems Theory*, 2(1):57–81, 1968. 1, 14

[Var03] Moshe Y. Vardi. Logic and Automata: A Match Made in Heaven. In Jos C. M. Baeten, Jan K. Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *Proceedings of the 30th International Colloquium on Automata, Languages and Programming, ICALP 2003*, volume 2719 of *Lecture Notes in Computer Science*, pages 64–65. Springer, 2003. 1

[Wal02] Igor Walukiewicz. Monadic Second-Order Logic on Tree-Like Structures. *Theoretical Computer Science*, 275(1-2):311–346, 2002. 4, 43, 76