Languages, Automata, and Logic

Wolfgang Thomas*

May 1996

Bericht 9607
Institut für Informatik und Praktische Mathematik
der Christian-Albrechts-Universität zu Kiel
D-24098 Kiel

*E-Mail: wt@informatik.uni-kiel.de
Work supported by ESPRIT BRA Working Group No. 6317 ASMICS 2 ("Algebraic and Syntactic Methods in Computer Science") and Deutsche Forschungsgemeinschaft (DFG Th 352/3-1).

Abstract

This paper is a survey on logical aspects of finite automata. Central points are the connection between finite automata and monadic second-order logic, the Ehrenfeucht-Fraïssé technique in the context of formal language theory, finite automata on ω-words and their determinization, and a self-contained proof of the "Rabin Tree Theorem".

Sections 5 and 6 contain material presented in a lecture series to the "Final Winter School of AMICS" (Palermo, February 1996). A modified version of the paper will be a chapter of the "Handbook of Formal Language Theory", edited by G. Rozenberg and A. Salomaa, to appear in Springer-Verlag.

Keywords: Finite automata, monadic second-order logic, first-order logic, regular languages, star-free languages, tree automata, Ehrenfeucht-Fraïssé game, ω-automata, temporal logic, Büchi automata, Rabin tree automata, determinacy, decidable theories.

Contents

1 Introduction

The subject of this chapter is the study of formal languages (mostly languages recognizable by finite automata) in the framework of mathematical logic.

The connection between automata and logic goes back to work of Büchi [Bü60] and Elgot [Elg61], who showed that finite automata and monadic second-order logic (interpreted over finite words) have the same expressive power, and that the transformations from formulas to automata and vice versa are effective. Later, in work of Büchi [Bü62], McNaughton [McN66], and Rabin [Rab69], such an equivalence was shown also between finite automata and monadic second-order logic over infinite words and trees. This research was initiated by decision problems for restricted systems of arithmetic and the problem of synthesizing circuits with nonterminating behaviour from logic specifications ([Chu63], [TB73]). The reduction of formulas to finite automata was the key to the solution of both problems: The monadic second-order theories S1S and S2S of one, respectively two successor functions were shown to be decidable in [Bü62] and [Rab69], leading to decidability results also for other interesting mathematical theories and for several logics of programs. Furthermore, it turned out (in the work of Büchi and Landweber [BL69]) that the circuit synthesis problem with respect to S1S-specifications is solvable effectively, which gave a new perspective to the automatic construction of nonterminating programs.

In the eighties, the bridge between the descriptive formalism of monadic second-order logic and the computational (or operational) model of finite automaton was refined and extended to allow practical use. Temporal logics and fixed-point logics took the role of the specification languages (replacing the classical systems of first-order logic and monadic second-order logic), and more efficient transformations from logic formulas to automata were found. This led to powerful algorithms and software systems for the verification of finite-state programs (*"model-checking"*). The area has developed into an own subject, built on an extensive literature which cannot be covered here in detail; as recent monographs in the field we mention [McM93], [Arn94a], and [Kur94].

The equivalence between automata and logical formalisms also started new tracks of research in language theory itself. For example, the classification theory of formal languages was deepened by including logical notions and techniques, and the logical approach helped in generalizing language theoretical results from the domain of words to more general structures like trees and partial orders.

The logical description of the behaviour of computational models was also taken up in complexity theory. Starting from Fagin's work [Fag74], it was shown that many complexity classes, such as NP, P, PSPACE, could be characterized by different versions of second-order logic (involving, for example, fixed point operators or transitive closure operators). This theory now forms the core of the subject *finite model theory* or (more specifically) *descriptive complexity theory*, and we refer the reader to [EF95] for a recent and comprehensive exposition.

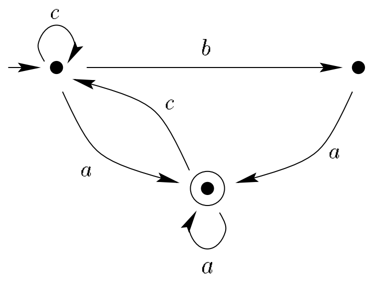
The topic of the present chapter, where finite automata are considered rather than resource-bounded Turing machines, may be called a *descriptive theory of recognizability*. In the logical framework, this corresponds to restricting second-order logic (as used in describing classical complexity classes) to its monadic (or even first-order) fragment.

A surprising merge of techniques and results from automata theory, logic, and complexity was finally achieved in *circuit complexity theory*, where the computational power of boolean circuits is studied, regarding restrictions in their size, depth, and types of allowed gates. It turned out that natural families of circuits (given by such bounds on size and depth) can be described by generalized models of finite automata as well as by appropriate systems of first-order logic. In Straubing's book [Str94] these results are developed in detail, including algebraic aspects (concerning, e.g., varieties of monoids associated with regular languages).

The main objective of this survey is to explain the precise relation between finite automata and monadic second-order logic and to give self-contained proofs of some fundamental results. This will include certain difficult automata theoretic constructions over infinite words and trees, e.g. Safra's determinization of ω-automata [Saf88] and Rabin's Tree Theorem [Rab69], which are as yet not accessible in textbooks or surveys, as well as a short exposition of the Ehrenfeucht-Fraïssé game technique and some of its applications concerning first-order logic in formal language theory. Thus, some complementary material to the related survey paper [Th90] is given. On the other hand, only short remarks will be made on the neighbour subjects mentioned above, for which the reader can refer to the cited monographs.

2 Models and Formulas

Let us start with a simple example to explain the description of formal languages by logical formulas. The finite automaton



accepts those words over the alphabet $A = \{a, b, c\}$ where no a is succeeded by a b, any b is succeeded by a, and a is the last letter. These three conditions can be expressed by a first-order formula, using variables $x, y, \ldots$ for letter positions, a formula $S(x,y)$ to indicate that position y succeeds x, and $Q_a(x)$ to formalize that position x carries letter a:

$$\begin{aligned}\varphi_1: \quad & \neg\exists x\exists y(S(x,y) \wedge Q_a(x) \wedge Q_b(y)) \ \wedge \ \forall x(Q_b(x) \rightarrow \exists y(S(x,y) \wedge Q_a(y))) \\ & \wedge \ \exists x(\neg\exists y S(x,y) \wedge Q_a(x))\end{aligned}$$

Note that $\neg\exists y S(x,y)$ expresses that x is the last letter position of the word under consideration.

Another example shows that variables $X, Y, \ldots$ ranging over *sets* of positions (and corresponding atomic formulas $X(y)$, meaning "$y \in X$") can be useful. Consider the set of words over $A = \{a, b\}$ where any two occurrences of b (such that no further b occurs between them) are separated by a block of an odd number of letters a. It suffices to express that for any two occurrences of b without a further b between them there is a set of positions containing the position of the first b, then every second position, and finally the position of the next b:

$$\begin{aligned}\varphi_2: \quad & \forall x\forall y(Q_b(x) \wedge x < y \wedge Q_b(y) \wedge \forall z(x < z \wedge z < y \rightarrow \neg Q_b(z)) \\ & \rightarrow \ \exists X(X(x) \wedge \forall u\forall v(S(u,v) \rightarrow (X(u) \leftrightarrow \neg X(v))) \wedge X(y)))\end{aligned}$$

In the remainder of the section we introduce the framework for the definition of formal languages more precisely. We include also more general structures than words, in particular labelled trees and graphs.

2.1 Words, Trees, and Graphs as Models

Let A be a finite alphabet and let $w = a_0 \ldots a_{n-1}$ be a word over A. The word w is represented by the relational structure

$$\underline{w} = (\mathrm{dom}(w), S^w, <^w, (Q_a^w)_{a \in A})$$

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

This framework is easily adapted to ω-words over a given alphabet A, i.e., to sequences $\alpha = a_0 a_1 \ldots$ with $a_i \in A$. The corresponding structures $\underline{\alpha}$ are of the form

$$\underline{\alpha} = (\omega, S^\alpha, <^\alpha, (Q_a^\alpha)_{a \in A})$$

where the domain is fixed as the set $\omega = \{0, 1, 2, \ldots\}$ of natural numbers.

Another generalization is to include trees. We shall restrict ourselves to proper binary trees, in which each node is either a leaf or has two successors (being ordered as left and right successor). This saves notation but covers all typical features arising with trees. Thus, nodes of trees will be represented as finite words over the alphabet $\{0,1\}$ (where 0 means "branch left" and 1 means "branch right"), and tree domains will be prefix closed subsets P of $\{0,1\}^*$, such that for any word $w \in P$ either both or none of $w0, w1$ also belong P.

A tree over the alphabet A is a map $t : \mathrm{dom}(t) \to A$ where $\mathrm{dom}(t)$ is a tree domain. The corresponding relational structure has the form

$$\underline{t} = (\mathrm{dom}(t), S_0^t, S_1^t, <^t, (Q_a^t)_{a\in A}).$$

Here S_0^t, S_1^t are the left, respectively right successor relations over $\mathrm{dom}(t)$ (with $(u, u0) \in S_0^t$ and $(u, u1) \in S_1^t$ for $u, u0, u1 \in \mathrm{dom}(t)$), $<^t$ is the proper prefix relation over d$om(t)$, and $Q_a^t = \{u \in \mathrm{dom}(t) \mid t(u) = a\}$. We say that a tree is finite if its domain is finite; as infinite trees over A we shall consider only the *full* binary trees, i.e., maps from $\{0,1\}^*$ to A. We denote by T_A the set of finite trees over A, and by T_A^ω the set of infinite (full binary) trees over A.

A further step of generalization is to consider vertex- and edge-labelled directed graphs. Usually, the vertex labels will be from an alphabet A, and the edge labels from an alphabet B. The vertex set is partitioned into sets Q_a (collecting the vertices with label a, respectively), and the edge set is partitioned into sets E_b (collecting the edges labelled b, respectively). Thus, graphs will be represented in the form

$$G = (V, (E_b^G)_{b\in B}, (Q_a^G)_{a\in A}),$$

where the Q_a^G are disjoint sets with $\bigcup_{a\in A} Q_a^G = V$ and the E_b^G are disjoint subsets of $V \times V$. In acyclic graphs, a partial order (the reflexive transitive closure of $E := \bigcup_{b\in B} E_b^G$) may be added. Tree models and word models arise then as special cases: For trees, V is a tree domain and there are two labels on edges, indicating transition to left and right successor; for words, there is only one label for the edge relation (which coincides with the successor relation).

When no confusion arises we cancel the superscripts w, α, t, G for the relations and just speak, for instance, of the successor relation S or the ordering $<$.

Two versions of graphs which are important in a generalized theory of formal languages are *Mazurkiewicz trace graphs* ([DR95]) and *texts* ([ER93]). Trace graphs arise from words by a "dependence relation" on the alphabet, and texts are obtained from words by introducing a second (arbitrary) successor relation. More details will be given later in this chapter in connection with results related to these structures.

2.2 First-Order Logic

Properties of words, trees, or graphs can be formalized in logical languages. We begin with the *first-order language*.

Consider word models over the alphabet A. The corresponding first-order language has variables $x, y, \dots$ ranging over positions in word models, and is built up from atomic formulas of the form

$$x = y, \quad S(x,y), \quad x < y, \quad Q_a(x) \text{ for } a \in A$$

by means of the connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ and the quantifiers $\exists$ and $\forall$. The set of used relation symbols $S, <, Q_a$ is called the *signature* of this first-order language. (The equality sign $=$ is tacitly assumed present.) The notation $\varphi(x_1, \dots, x_n)$ indicates that in the formula φ at most the variables $x_1, \dots, x_n$ occur free (i.e., not in the scope of some quantifier). A *sentence* is a formula without free variables.

If $p_1, \dots, p_n$ are positions from $\mathrm{dom}(w)$, then $(\underline{w}, p_1, \dots, p_n) \models \varphi(x_1, \dots, x_n)$ means that φ is satisfied in $\underline{w}$ when $=, S, <, Q_a$ are interpreted by equality, $S^w, <^w, Q_a^w$, respectively, and $p_1, \dots, p_n$ serve as interpretations of $x_1, \dots, x_n$, respectively. The empty model is usually excluded in the framework of mathematical logic. In the sequel we allow the empty word ϵ as member of formal languages, and admit the empty model as interpretation of sentences. The natural convention that $\underline{\epsilon}$ satisfies universal sentences $\forall x \varphi(x)$ but does not satisfy existential sentences $\exists x \varphi(x)$ fixes the satisfaction relation between $\underline{\epsilon}$ and sentences φ.

The language defined by the sentence φ is $L(\varphi) = \{w \in A^* \mid \underline{w} \models \varphi\}$. Similarly, the ω-language defined by φ is $L_\omega(\varphi) = \{\alpha \in A^\omega \mid \underline{\alpha} \models \varphi\}$. For the example sentence φ_2 of the introduction of this section, $L_\omega(\varphi_2)$ contains all ω-words over $\{a, b\}$ where between any two (successive) occurrences of b there is an odd number of letters a.

We say that a language $L \subseteq A^*$ (resp. ω-language $L \subseteq A^\omega$) is $\mathrm{FO}[S,<]$-definable (or first-order definable) if a first-order sentence φ as above exists with $L = L(\varphi)$ (resp. with $L = L_\omega(\varphi)$). Similarly, by $\mathrm{FO}[S]$-definability we mean the existence of such a sentence in which no use is made of $<$. Note that in the first case we may as well drop the symbol S for successor since $S(x,y)$ can be expressed in terms of $<$ by the formula $x < y \wedge \neg\exists z(x < z \wedge z < y)$.

In the definition of word properties, it is often convenient to allow predicates $\mathrm{first}(x)$ and $\mathrm{last}(x)$ which apply only to the first, respectively last position (if it exists) of a word model. Thus, $\mathrm{first}(x)$, $\mathrm{last}(x)$ will stand for the formulas $\neg\exists y S(y,x)$ and $\neg\exists y S(x,y)$, respectively. (If $<$ is used, replace S by $<$.) If only nonempty words are considered, there is the alternative to introduce special constants min and max denoting the first, respectively last element of a word model (which exist by the non-emptiness assumption).

In an analogous way, first-order formulas over tree models and graph models are introduced. The signature is adapted accordingly, along with the interpretation of its relation symbols. So for (binary) trees we use the relation symbols S_1, S_2 for the two successor relations, and $<$ stands now for the partial order of the proper prefix relation over tree domains. By $T(\varphi)$, resp. $T_\omega(\varphi)$, we denote the set of finite, resp. infinite trees (over a given alphabet A) which satisfy the sentence φ.

Sometimes it is convenient to use function symbols rather than relation symbols, for example the symbols suc, suc_0, suc_1 for successor functions instead of S, S_0, S_1 as introduced above. This allows shorter formalizations especially if compositions of functions are considered. For example, one can write $y = \mathrm{suc}(\mathrm{suc}(\mathrm{suc}(x)))$ instead of $\exists z_1 \exists z_2 (S(x, z_1) \wedge S(z_1, z_2) \wedge S(z_2, y))$. However, our general considerations would become more complicated with function symbols, e.g. a convention would be necessary for the assignment of a successor to the last position of a word (or alternatively, partial functions would have to be admitted). Since it is always possible to eliminate function symbols in terms of relation symbols (for the graphs of the functions under consideration), we shall restrict to the relational case in the sequel.

Over graphs, the edge relation symbols E_b take the role of the successor relation symbols S, S_0, S_1. Thus, given the label alphabets A (for vertices) and B (for edges), the atomic formulas of the associated first-order language are $x = y$, $E_b(x, y)$, and $Q_a(x)$.

We shall use some standard results of quantifier logic, especially the *prenex normal form* into which each (first-order) formula can be transformed. Here a prefix of quantifiers precedes a quantifier-free kernel. If successive quantifiers of the same type are grouped into n blocks, beginning say with existential quantifiers, such a formula has the form

$$\exists \overline{x}_1 \forall \overline{x}_2 \dots \exists / \forall \overline{x}_n \varphi_0(\overline{x}_1, \overline{x}_2, \dots \overline{x}_n)$$

with tuples $\overline{x}_i$ of variables and quantifier-free φ_0. Such a formula is called a Σ^0_n*-formula*. By a Π^0_n*-formula* we mean the dual case, i.e., a formula where there are n alternating blocks of quantifiers beginning with a block of universal quantifiers. By the laws of quantifier logic, the negation of a Σ^0_n-formula can be written as a Π^0_n-formula. Boolean combinations of Σ^0_n-formulas will be called $B(\Sigma^0_n)$-formulas. The superscript 0 indicates that the classification according to first-order quantifiers is considered (and may be omitted if this context is clear); a superscript 1 refers to the classification by second-order quantifiers.

2.3 Monadic Second-Order Logic

We extend the logical formalism by second-order variables $X, Y, \dots, X_1, \dots$ which range over sets of elements of models, i.e. sets of letter positions, sets of tree nodes, sets of graph vertices. Corresponding atomic formulas $X(x)$, $X(y), \dots$ are also introduced, with the intended meaning "x belongs to X", "y belongs to X", etc. Since sets are "monadic second-order objects" , in contrast to relations of higher arity (which are "polyadic"), the resulting system is called *monadic second-order logic*, short MSO-logic.

Again, for second-order formulas a prenex normal form exists. Here one can shift all second-order quantifiers in front of first-order quantifiers, by taking singletons as representations of elements. For example, $\forall x \exists Y \varphi(x, Y)$ is equivalent to $\forall X \exists Y \forall u \forall v \forall x((X(u) \wedge X(v) \to u = v) \wedge (X(x) \to \varphi(x, Y)))$. Now a Σ^1_n-formula is a formula where a prefix of n second-order quantifier blocks, starting with existential quantifiers, precedes a formula where at most first-order quantifiers occur. Σ^1_1-formulas of MSO-logic are also called *existential monadic second-order formulas*, short EMSO-formulas.

Note that in MSO-logic the order relation $<$ over words becomes definable in terms of successor S: We have (over word models) the equivalence between $x < y$ and

$$\neg x = y \wedge \forall X(X(x) \wedge \forall z \forall z'(X(z) \wedge S(z, z') \to X(z')) \to X(y)).$$

We obtain that any FO[<]-definable word language is also MSO[S]-definable (and henceforth we just say "MSO-definable"). Over trees, a similar definition in terms of S_0, S_1 can be given for the partial tree order $<$ (the proper prefix relation over $\text{dom}(t)$ for a given tree t).

In the study of monadic second-order logic we shall use a modified logical system of same expressive power, which we call MSO_0-logic. It has a simpler syntax, in which the first-order variables are cancelled. As for the prenex normal form of MSO-formulas, the idea is to simulate (quantifiers over) elements by (quantifiers over) singletons. Thus $\{x\} \subseteq X$ will replace $X(x)$. There are new atomic formulas in MSO_0-logic, namely (given the label alphabet A)

$$X \subseteq Y, \quad \text{Sing}(X), \quad \text{Suc}(X, Y), \quad X \subseteq Q_a \text{ (for } a \in A)$$

meaning that X is a subset of Y, X is a singleton, X, Y are singletons $\{x\}$, $\{y\}$ with $S(x, y)$, and X is a subset of Q_a, respectively.

The translation from MSO- to MSO_0-logic is easy by induction over the construction of MSO-formulas. For example,

$$\forall x(Q_a(x) \to \exists y(S(x, y) \wedge Z(y)))$$

is rewritten as

$$\forall X(\text{Sing}(X) \wedge X \subseteq Q_a \to \exists Y(\text{Sing}(Y) \wedge \text{Suc}(X, Y) \wedge Y \subseteq Z)).$$

Over trees and graphs, we would use formulas $\text{Suc}_i(X, Y)$ and $E_b(X, Y)$ instead of $\text{Suc}(X, Y)$.

An MSO-formula $\varphi(X_1, \ldots, X_n)$ with at most the free variables $X_1, \ldots, X_n$ is interpreted in a word model (tree model, graph) with n designated subsets $P_1, \ldots, P_n$. Such a model represents a word (tree, graph) over the expanded alphabet $A' = A \times \{0, 1\}^n$, where the label $(a, c_1, \ldots, c_n)$ of position p (node p, vertex p) indicates that p carries label a from A and that p belongs to P_j iff $c_j = 1$. For instance, the ω-word model $(\underline{\alpha}, P_1, P_2)$ where $\alpha = abbaaaa\ldots$, P_1 is the set of even numbers, and P_2 is the set of prime numbers, will be identified with the following ω-word over $\{a, b\} \times \{0, 1\}^2$, where letters are written as columns:

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

In the sequel we shall often use such identification of set expansions of models with models over extended alphabets.

It is worth mentioning that in the logical framework there is no essential difficulty in transferring definability notions from the domain of words to the more extended domains of trees and graphs, and that also the transition from finite models to infinite models does not involve any conceptual problem. It is only necessary to adapt the signature under consideration and to change the class of admitted models. For definability notions from formal language theory, which are based on automata, grammars, or regular expressions, such generalizations are more involved, and sometimes only possible with additional conventions that need special justification. In this sense the logical approach may serve as a support and guideline for generalizing classical formal language theory.

3 Automata and MSO-Logic on Finite Words and Trees

3.1 MSO-Logic on Words

To specify recognizable (i.e., regular) languages, we refer to nondeterministic automata over an alphabet A, which are presented in the form $\mathcal{A} = (Q, A, q_0, \Delta, F)$ where Q is the finite state set, A is the input alphabet, q_0 the initial state, $\Delta \subseteq Q \times A \times Q$ the transition relation, and F the set of final states. A word $w = a_0 \dots a_{n-1}$ is accepted by $\mathcal{A}$ if there is a successful run of $\mathcal{A}$ on w, i.e. a sequence $\rho = \rho(0) \dots \rho(n)$ of states with $\rho(0) = q_0$, $(\rho(i), a_i, \rho(i+1)) \in \Delta$ for $i < n$, and $\rho(n) \in F$. The language recognized by $\mathcal{A}$ collects all words over A accepted by $\mathcal{A}$.

Theorem 3.1 (Büchi [Bü60], Elgot [Elg61]) *A language of finite words is recognizable by a finite automaton iff it is MSO$[S]$-definable, and both conversions, from automata to formulas and vice versa, are effective.*

Proof. To show the direction from left to right, let $\mathcal{A} = (Q, A, q_0, \Delta, F)$ be a finite automaton. Assume $Q = \{0, \dots, k\}$ where $q_0 = 0$. We have to find a monadic second-order sentence that expresses in any given word model $\underline{w}$ (over A) that $\mathcal{A}$ accepts w. Over a word $w = a_0 \dots a_{n-1}$, the sentence will state the existence of a successful run $p_0, \dots, p_n$ of $\mathcal{A}$, i.e. with $p_0 = 0$, $(p_i, q_i, p_{i+1}) \in \Delta$ for $i < n$, and $p_n \in F$. We may code such a state sequence up to p_{n-1} by a tuple $(X_0, \dots, X_k)$ of pairwise disjoint subsets of $\{0, \dots, n-1\}$ such that X_i contains those positions of w where state i is assumed. (A more efficient coding would use a correspondence between states and 0-1-vectors of suitable length, which allows to describe a run over 2^m states by an m-tuple of subsets of the word domain.) From the last state p_{n-1} the automaton should be able to reach a final state via the word's last letter a_{n-1}. Thus, $\mathcal{A}$ accepts the nonempty word w iff

$$\begin{array}{lll} \underline{w} \models & \exists X_0 \ldots \exists X_k & (\bigwedge_{i \neq j} \forall x \neg(X_i(x) \wedge X_j(x)) \\ & & \wedge \ \forall x(\mathrm{first}(x) \rightarrow X_0(x)) \\ & & \wedge \ \forall x \forall y(S(x,y) \rightarrow \bigvee_{(i,a,j) \in \Delta}(X_i(x) \wedge Q_a(x) \wedge X_j(y))) \\ & & \wedge \ \forall x(\mathrm{last}(x) \rightarrow \bigvee_{\exists j \in F:(i,a,j) \in \Delta}(X_i(x) \wedge Q_a(x))) \end{array}$$

The empty word satisfies this sentence (with $X_i = \emptyset$). Thus, if $\mathcal{A}$ does not accept ϵ, a corresponding clause (such as $\exists x \; x = x$) should be added.

Let us show the direction from right to left. Here we refer to the MSO_0-formulas introduced in the previous section and show the claim by induction on these formulas. We have to exhibit for any given formula $\varphi(X_1, \ldots, X_n)$ a finite automaton which accepts precisely those words $w \in A \times \{0,1\}^n$ which satisfy φ. (Recall that such words are represented by word models $(\underline{w}, P_1, \ldots, P_n)$.) It is easy to present finite automata that recognize the sets defined by atomic formulas $X_j \subseteq X_k$, $\mathrm{Sing}(X_j)$, $\mathrm{Suc}(X_j, X_k)$, and $X_j \subseteq Q_a$. E.g., the finite automaton checking whether $X_j \subseteq X_k$ holds in $w \in (A \times \{0,1\}^n)^*$ has to verify that whenever 1 occurs in the j-th additional 0-1-component it occurs also in the k-th additional 0-1-component.

For the inductive step, it suffices to consider the connectives $\neg$, $\vee$ and the existential set quantification, since the other connectives and the universal set quantifier are definable in terms of them. This in turn amounts to the proof that the class of recognizable languages shares well-known closure properties, namely closure under complement, under union, and under projection. Let us discuss the latter case: Assuming that the language defined by the formula $\psi(X_1, \ldots, X_n)$ over the alphabet $A \times \{0,1\}^n$ is recognized by the automaton $\mathcal{A}$, we have to exhibit an automaton corresponding to the formula $\varphi(X_1, \ldots, X_{n-1}) = \exists X_n \psi(X_1, \ldots, X_n)$. The required automaton (over $A \times \{0,1\}^{n-1}$) just has to guess (by nondeterminism) a 0-1-sequence which should define the n-th additional components and has to work on this extended word over $A \times \{0,1\}^n$ like $\mathcal{A}$. $\square$

The formula in the above proof, describing acceptance of the underlying word model by an automaton, is an EMSO-formula of a special type. Invoking the second part of the proof, we see that it provides a normal form of MSO[S]-formulas, in Büchi's terminology an "automata normal form".

Corollary 3.2 *Any MSO[S]-formula can be written as an EMSO[S]-formula.*

In [Th82a] it is shown that even a single existential set quantifier suffices in EMSO-formulas for defining recognizable languages.

The automata theoretic approach to monadic second-order logic yields a form of quantifier elimination, as is visible from the reduction of arbitrary formulas to the mentioned automata normal form. As in classical logic, one derives from it decidability results. Not all quantifiers are eliminated here, but a normal form is reached in which only existential set quantifiers and (in the kernel) universal first-order quantifiers appear. (If the predicates "first" and "last" were expanded, one first-order quantifier alternation would have to be added.) Advantages of the normal form are its strong closure properties (under boolean operations and projection) and its algorithmic (or operational) meaning.

The potential to eliminate quantifiers rests on the simultaneous closure of the class of recognizable languages under projection (corresponding to existential quantification) and complement (allowing to treat the dual, universal quantification). In the automata theoretic framework, this is usually shown via the reduction of nondeterministic automata (which yield projection easily) to deterministic automata (which yield complementation easily). Successive alternations of quantifiers thus amount to successive applications of the powerset construction for automata. This means that (in the straightforward approach) each quantifier alternation induces an exponential blow-up of the size of corresponding finite automata. Indeed, from results of Meyer and Stockmeyer (see [AHU74]) it follows that, regarding computation time, a blow-up cannot be avoided: The time complexity of any algorithm converting MSO-formulas (even FO$[S,<]$-formulas) to equivalent finite automata cannot be bounded by an elementary function (i.e. by an iterated exponential of the form $2\hat{}(2\hat{}\dots(2^n)\dots)$ in the length n of the given formula). It is remarkable, however, that a conversion algorithm has been implemented which allows nontrivial practical applications in hardware verification ([BK95]).

In a corresponding reduction of MSO-logic to finite automata over infinite words and infinite trees, the determinization and complementation results are more difficult; this will be treated in Sections 5 and 6.

A natural generalization of MSO-logic is to admit second-order variables of higher arity, i.e. variables ranging over relations, together with quantifiers for them. This leads to a much larger language class than the class of regular languages:

Theorem 3.3 (Fagin [Fag74]) *A language belongs to the complexity class NP iff it is definable in (general) existential second-order logic.*

It follows that full second-order logic covers all languages in the polynomial time hierarchy. Other second-order concepts, such as least fixed-point operator or transitive closure operator, lead to logics which characterize further complexity classes like P, NLOGSPACE, PSPACE. For these results of descriptive complexity theory the reader should consult [EF95].

If only binary relations are admitted and restricted to so-called "matchings" ([LST95]), a characterization of the context-free languages is obtained. A relation

$R \subseteq \{0, \dots, n-1\}^2$ is called *matching* if it contains only pairs (i, j) with $i < j$ such that each position i belongs to at most one pair in R, and there are no "crossings" between pairs (i.e., for (i, j), $(k, l) \in R$, $i < k < j$ implies $i < k < l < j$). Typically, this kind of binary relation serves to define Dyck languages, by connecting positions where matching letters $a_i, \overline{a}_i$ occur.

Theorem 3.4 (Lautemann, Schwentick, Thérien [LST95]) *A language is context-free iff it is definable in existential second-order logic where the second-order variables range only over matchings.*

We turn to some applications of Theorem 3.1 to decision problems and results concerning definability of sets and relations over finite words. Büchi and Elgot used the result to derive the decidability of the *weak monadic second-order theory of successor*, sometimes denoted WS1S; it consists of all MSO-sentences which are true in the structure $(\omega, S, <)$ under the provision that set quantifiers range only over finite sets. Indeed, any MSO-sentence φ with this interpretation is equivalent to an input-free finite automaton on finite words, and truth of φ in $(\omega, S, <)$ amounts to the existence of a successful run of this automaton (which is easily checked).

In [Bü60] and [Elg61] it was also noted that from the decidability of WS1S the decidability of *Presburger Arithmetic* can be inferred, the set of true first-order sentences in the structure $(\omega, +)$. The idea is to represent numbers in binary, i.e. as 0-1-words, and to view any 0-1-word as (characteristic function of) a finite set. It is convenient to write down the binary representations in reversed order, which puts the i-th bit b_i in the expansion $\Sigma_{i=0}^{l} b_i 2^i$ to position i, yielding the word $b_0 \dots b_l$. Then, for example, the number 25 with reversed binary representation 10011 corresponds to the finite set $\{0, 3, 4\}$. It is now easy to write down a formula $\varphi(X_1, X_2, X_3)$ which expresses that the (finite) sets X_1, X_2, X_3 represent numbers x_1, x_2, x_3 such that $x_1 + x_2 = x_3$: One describes the addition algorithm which proceeds digit by digit (using successor to proceed to the next digit and the existence of an auxiliary set for the carries). In this way, any first-order formula $\varphi(x_1, \dots, x_n)$ in the signature $+$ is inductively transformed into a corresponding weak MSO-formula $\varphi'(X_1, \dots, X_n)$, using finite-set quantifiers in place of first-order quantifiers over numbers. The decidability of Presburger arithmetic follows by applying this translation to first-order sentences in the signature $+$ (without free variables) and invoking decidability of WS1S.

Instead of translating Presburger formulas $\varphi(x_1, \dots, x_n)$ into weak MSO-logic one can proceed directly to finite automata. An input for such an automaton is a word over the alphabet $\{0, 1\}^n$ which stands for an n-tuple $(k_1, \dots k_n)$ of numbers; the sequence of the j-th components is the reversed binary representation of k_j. The length of the word is determined by the highest digit carrying a 1; if this highest nonzero digit, say at position l, occurs with k_j, the representations of the k_m with $m \neq j$ are filled from their highest nonzero digit with zeroes up to this position l. A finite automaton which scans such a word over $\{0, 1\}^n$ can be viewed as an acceptor with one reading head per component, whose heads move synchronously through the input. Thus one calls word relations recognized by such automata *synchronized rational relations* ([FS93]). In the context of numbers represented over base 2, one speaks of *2-recognizable relations* of natural numbers.

In Büchi's work [Bü60], the question was considered which extension of Presburger arithmetic would allow to define precisely the 2-recognizable sets of natural numbers. In other words, how can one extend Presburger arithmetic to have also a translation back into weak MSO-logic (or into finite automata)? Büchi suggested to adjoin the predicate of being a power of 2, but it turned out that slightly stronger arithmetical means are necessary, and in fact that the function V_2 is appropriate which associates with each number $m > 0$ the greatest power of 2 which divides m.

In general, one considers p-ary representations of natural numbers for any $p > 0$ and the associated notion of a *p-recognizable relation*, using automata working over the alphabet $\{0, \ldots, p-1\}^n$ if the relation is n-ary. (Then the 1-recognizable sets of numbers are the ultimately periodic ones.) On the logical side, one defines for $p > 0$ the function V_p by

$$V_p(m) = \text{greatest power of } p \text{ which divides } m$$

for $m > 0$. Then the following equivalence result holds:

Theorem 3.5 (cf. [BHMV94])
A relation of natural numbers is p-recognizable iff it is first-order definable in the structure $(\omega, +, V_p)$ (by a formula $\varphi(x_1, \ldots, x_n)$ if the relation is n-ary).

A deep theorem due to Cobham connects the notions of p- and q-recognizability for different p and q: A *set* of natural numbers which is both p- and q-recognizable for multiplicatively independent p and q must be 1-recognizable, hence ultimately periodic (see e.g. [Per90]). Here two numbers p, q are called multiplicatively independent if there are no powers p^m and q^n $(m, n > 0)$ which coincide. A generalization of Cobham's Theorem, namely for relations instead of sets of numbers, was obtained by Semenov and later given a very elegant proof by Muchnik; for comprehensive expositions see the lucid survey [BHMV94] or [MV96].

It is interesting to note that the expressive power of finite automata which recognize relations in an asynchronous manner (such that the reading heads on different components may be apart by arbitrary distances) is much greater than in the synchronous case. For instance, while the class of synchronized rational relations is captured by the weak MSO-logic of successor and thus closed under boolean operations and projection, the application of boolean operations and projection to asynchronously recognized relations leads to nonrecursive relations, and indeed one can exhaust in this way the arithmetical hierarchy of word-relations and languages ([Sei92]). On the other hand, if the distance between the reading heads is uniformly bounded, a reduction to synchronized mode is possible, even over infinite input words ([FS93]).

3.2 MSO-Logic on Traces and Trees

The mathematical core of Theorem 3.1 is the fact that the model of finite automaton is closed under the operations of complementation and projection, or, in logical terms, negation and existential quantification. It is natural to ask over which generalized structures (instead of finite words) a similar theory of finite automata is possible, aiming at corresponding logical consequences.

In this section we discuss some basic classes of structures where such a generalization is possible.

The first is the class of (dependency graphs of) *Mazurkiewicz traces*, cf. [DR95], [DM96]. Traces are formed over a *dependence alphabet*, which is a pair (A, D) with an alphabet A and a reflexive and symmetric "dependency relation" $D \subseteq A \times A$. Note that each letter is considered dependent on itself. We view traces as special acyclic (and hence partially ordered) graphs whose vertices are labelled in A and whose edge relation respects D in the sense that edges connect only vertices carrying dependent letters and that any two vertices labelled by dependent letters are connected by a path. Thus, by reflexivity of D, an antichain in a trace graph (i.e., a set of vertices which are pairwise unrelated by the partial order) can have at most $|A|$ elements. In order to obtain a canonical representation, we keep in the edge set E only those edges that are present in the Hasse diagram of the partial order (i.e., not generated by transitive closure from other edges), and also include the generated partial order $<$. Thus a *trace graph* has the form $(V, E, <, (Q_{a \in A}))$, such that the above-mentioned conditions on vertices with dependent letters are satisfied. A trace language is identified with a set of trace graphs over the given dependency alphabet (A, D). The notion of MSO-definability for trace languages is now canonical.

On the other hand, it is nontrivial to set up a model of finite automaton which works in accordance with the idea of dependency and independency inherent in the definition of traces over an alphabet (A, D). Zielonka suggested in [Zie87] the model of *asynchronous finite automaton*. The idea is to decompose the dependency alphabet into (possibly overlapping) maximal cliques w.r.t. the dependence relation D; each such clique is called a "process". (For example, if $A = \{a, b, c, d\}$ and the dependency relation is generated by the pairs (a, b), (b, c), (c, d), then these three pairs form three processes.) A run of an asynchronous automaton on a trace is fixed by associating a number of states to each vertex: if the vertex is labelled a then one state for each process to which letter a belongs is listed. The transition relation now defines which state assignments for a vertex are possible, taking into account its label a and the state assignments of the last occurrences of vertices (in the partial order) where processes of a were involved.

In deterministic asynchronous automata the state assignment is uniquely determined by the state assignments of preceding vertices. An initial condition for first vertices (with respect to $<$) and a final condition for the last ones is also given. It turns out that recognizability by asynchronous automata also matches the algebraic definition of recognizability and thus provides a robust and natural notion (cf. [DR95]). The fundamental result on asynchronous automata states that the nondeterministic and the deterministic version are expressively equivalent ([Zie87]). Thus, the proof method of Theorem 3.1 can be applied, and one obtains the following result, shown e.g. in [DR95]:

Theorem 3.6 *A set of traces is recognizable by an asynchronous automaton iff it is MSO-definable.*

For certain *rational trace languages*, which transcend the class of recognizable trace languages, Choffrut and Guerra [CG95] found a logical characterization, extending MSO-logic with formulas which allow to compare cardinalities of sets.

Over trees, the situation is somewhat easier, referring to the theory of finite tree automata (see e.g. [GS84] or the chapter on tree languages in this Handbook). We consider tree automata working in bottom-up (or frontier-to-root) mode. In their transition relation, they can at each node only merge information which is provided by the states assumed at the son nodes. There are no points where information has to be kept along diverging branches which later may join again (as it may happen in trace graphs).

Definition 3.7 A *tree automaton* has the form $\mathcal{A} = (Q, A, q_0, \Delta, F)$ where Q is finite, $q_0 \in Q$, $F \subseteq Q$, and $\Delta \subseteq Q \times A \times Q \times Q$; a transition (q, a, q', q'') allows to proceed from two states q', q'' at the successor nodes of a node u to state q at u while reading letter a as label of u. A run of $\mathcal{A}$ on an input tree t is built up in the canonical way as a map $\rho : \mathrm{dom}(t) \to Q$, initialized for any leaf u labelled a using a transition (q, a, q_0, q_0) (which leads to the assignment $\rho(u) = q$). The run ρ is called successful if $\rho(\epsilon) \in F$, and a tree is accepted if a successful run exists on it. The tree language recognized by $\mathcal{A}$ consists of the trees accepted by $\mathcal{A}$.

The classical subset construction works without essential change also for tree automata of this form, which shows that over trees the nondeterministic and the deterministic automaton model (in frontier-to-root mode) are equivalent. So the method of Theorem 3.1 can be applied again:

Theorem 3.8 (Thatcher and Wright [TW68], Doner [Don70]) *A set of finite trees is recognizable by a finite tree automaton iff it is MSO-definable.*

With the same argument as for MSO-logic on words, also over finite trees MSO-logic is equivalent in expressive power to EMSO-logic.

For the proof of Theorem 3.8, seemingly weaker quantifiers than those ranging over arbitrary subsets of tree domains suffice. Let us call *antichain* a subset P of a tree domain such that any two distinct nodes in P are incomparable by the prefix relation $<$ (i.e. they do not belong to a common path). *Antichain logic* is the restriction of monadic second-order logic where set quantifications range over antichains only. Now we note that over proper binary trees (where each node has either two successors or is a leaf), the inner nodes can be mapped injectively into the set of leaves: From a given inner node we follow the path which first branches right and then always branches left until a leaf is reached. Thus a set of inner nodes can be coded by a set of leaves and hence by an antichain. Using this idea, quantifiers over subsets of proper binary trees can be simulated by quantifiers over antichains:

Proposition 3.9 ([PT93]) *A set of proper trees (without unary branching) is recognizable by a finite tree automaton iff it is definable in antichain logic.*

Similarly, *chain logic* is introduced; it allows only set quantifiers ranging over sets where any two elements are related via the partial prefix order. As shown in [Th84b], this system is strictly weaker than MSO-logic.

Theorem 3.8 allows to obtain decidability results for tree theories, as Theorem 3.1 does for theories of successor (i.e, fragments of arithmetic). In [TW68] and [Don70], the weak monadic theory of the binary infinite tree was shown to be decidable, using the decidability of the emptiness problem for tree automata.

Dauchet and Tison [DT90] applied tree automata in the spirit of the decidability proof for Presburger arithmetic (as discussed in the previous section). Here an n-ary relation of finite trees with label alphabet A is captured by a set of trees over the alphabet A^n (possibly extended by a dummy label in the individual components if tuples of trees with different domains are to be handled). In analogy to the case of word relations, the j-th components code the j-th tree of the n-tuple. Three relations between trees are considered in [DT90], each of them given by a finite tree rewriting system S ("ground rewriting system"): The first relation R_1 collects all tree pairs (s,t) such that t is obtained from s by application of a rule from S, the second relation R_2 contains all pairs (s,t) where such rewriting steps are applied in parallel, and the third, R_3, is the transitive closure of R_1. Now the first-order theory of the ground rewrite system S is defined to be the set of all first-order sentences in the signature with the relations R_1, R_2, R_3 which are true about the domain of all trees over A with the relations R_i determined by S as explained above.

Inductively, each first-order formula $\varphi(x_1, \ldots, x_n)$ of this language can be transformed into a tree automaton accepting those n-tuples of trees which satisfy φ. Hence the following result is obtained:

Theorem 3.10 ([DT90]) *The first-order theory of any ground rewrite system is decidable.*

An interesting issue in present research is the problem of finding more general domains of graphs where the automata theoretic approach to MSO-logic works again. These attempts are subject to limitations, however, for instance regarding decidability results: There are natural classes of acyclic labelled graphs over which even the satisfiability of EMSO-formulas is undecidable. The most basic example is the class of "pictures" or "two-dimensional words", i.e. rectangular arrays of labelled vertices which are connected by two successor relations, a horizontal and a vertical one. It is easy to show that the halting problem for Turing machines is reducible to satisfiability of EMSO-formulas over such pictures: For each Turing machine $\mathcal{M}$ one can write down an EMSO-sentence $\varphi_{\mathcal{M}}$, describing those pictures that code a halting computation of $\mathcal{M}$ starting with the empty tape. (Such a picture, say over the alphabet $\{0,1\}^k$, represents a halting computation in the form of a two-dimensional space-time-diagram, such that all points visited by $\mathcal{M}$ belong to the picture. Existential quantifiers over sets $X_1, \dots, X_k$ serve to express that an appropriate assignment of values from $\{0,1\}^k$ to picture points exists, while the local conditions on neighbour letters, as fixed by the Turing machine instructions, are expressible in first-order logic.) Thus, satisfiability of $\varphi_{\mathcal{M}}$ in the domain of pictures amounts to existence of a halting computation of $\mathcal{M}$ starting on the empty tape, whence satisfiability of EMSO-sentences over pictures is undecidable. For a detailed discussion of picture languages see [GR96].

Over the class of pictures, also other facts fail which are familiar from the domain of words or trees. First, EMSO-logic turns out to be strictly weaker than MSO-logic over pictures; for example, the set of $(n \times 2n)$-dimensional pictures which are composed from two identical square pictures is MSO-definable but not EMSO-definable (cf. [GRST96]). Closely related is the fact that the powerset construction fails for canonical models of finite automata over pictures and acyclic graphs (see [PST94], where a claim of [KS81] concerning applicability of the powerset construction is corrected). Also in the domain of arbitrary finite graphs MSO-logic can be separated from EMSO-logic: Connectivity is an MSO-expressible graph property which is not definable in EMSO-logic (cf. [FSV95]).

Some classes of graphs have been found where the classical technique of connecting MSO-logic with notions of recognizability can be applied; usually, however, this depends on the possibility to describe graph properties in terms of certain tree properties. As an example we mention properties of *texts*, a structure introduced in [ER93]. A text is a word which has a second ordering besides the natural ordering of letters. Alternatively, a text is presented as a word together with a permutation of its positions, for example in the form $(acabaacbc, (5,2,4,1,3,6,7,8,9))$. A text can be built up in a structured way, combining parts of it in the form of a tree structure, called *shape*. Hoogeboom and ten Pas showed in [HP94] that a natural algebraic notion of recognizability and definability in MSO-logic coincide for text languages where these tree representations involve only trees of bounded arity (i.e., can be handled by finite tree automata).

A more general framework of definability of graph properties (which implicitly involves terms and trees with unbounded arity) was developed by Courcelle [Cou90]. It is based on the construction of graphs in a many-sorted graph algebra and leads to an algebraic notion of recognizability of graph sets which is strictly more expressive than MSO-logic. The finitary framework of MSO-logic and tree automata is exceeded by the admission of infinitely many sorts in this graph algebra, a feature which is necessary, for example, in the definition of picture languages.

There is as yet no characterization of the classes of graphs which share the desirable properties of the classical theory, namely a decidable satisfiability problem (or validity problem) for MSO-formulas, the equivalence between MSO-logic and its existential part, EMSO-logic, as well as between MSO-logic and finite-state acceptors. An interesting class where this question is open is given by the graphs of *bounded tree-width*; see [Cou91] and [See92] for a detailed treatment and partial results in two complementary approaches.

A general method to construct sets of graphs from sets of trees is to apply *monadic second-order interpretations*. An interpretation describes a relational structure $\mathcal{S}$ (say, a graph) within a given structure $\mathcal{R}$ (say, a tree) by providing "defining formulas". One formula (with a free variable) defines a copy of the domain of $\mathcal{S}$ within $\mathcal{R}$, and further formulas are provided to define the relations of $\mathcal{S}$ as relations over that part of $\mathcal{R}$ which represents $\mathcal{S}$. Seese [See92] applies interpretations of graphs in trees and uses tree automata theory to obtain decidability results, as well as upper time-bounds for computational graph problems. In [Cou94], the related notion of *monadic second-order graph transduction* is studied. Sets of graphs which are generated by (different versions of) context-free graph grammars are shown to be presentable as images of recognizable tree languages under such MSO-definable graph transductions. A detailed exposition of these results and their applications is given in the survey [Cou96].

4 First-Order Definability

4.1 The Ehrenfeucht-Fraïssé Game

The limited expressive power of finite automata (and hence MSO-logic) over words or trees is verified conveniently using pumping lemmas and related combinatorial arguments. For first-order logic, the situation is more involved. The most versatile method to show non-definability in systems of first-order logic is the Ehrenfeucht-Fraïssé game, and it is applied in characterizations of several classes of first-order definable languages. We give the main facts in a brief overview; more background can be found e.g. in [EFT94] or [EF95].

In the sequel we consider a first-order language (with equality) for a signature *Sig* with the unary relation symbols $Q_1, \ldots, Q_k$ and the binary relation symbols

$R_1, \ldots, R_l$. The restriction to unary and binary relations is inessential but saves notation and is enough for the present purposes. Letters Q and R will indicate unary, resp. binary relation symbols. The structures for the signature Sig are of the form $\mathcal{S} = (S, Q_1^S, \ldots, Q_k^S, R_1^S, \ldots, R_l^S)$ where S is the structure's universe, $Q_i^S \subseteq S$ for $1 \leq i \leq k$ and $R_j^S \subseteq S \times S$ for $1 \leq j \leq l$. Sometimes we expand such a structure by designated elements from its universe. For example, if $\overline{s} = (s_1, \ldots, s_n)$ is an n-tuple of elements from S and $\varphi(\overline{x})$ is a formula where at most the variables of $\overline{x} = (x_1, \ldots, x_n)$ occur free, then $(\mathcal{S}, \overline{s}) \models \varphi(\overline{x})$ indicates that φ holds in $\mathcal{S}$ when interpreting x_i by s_i for $i = 1, \ldots, n$.

The *quantifier-depth* $qd(\varphi)$ of formulas φ is the maximal number of nested quantifiers in φ. Given $m \geq 0$, two structures $\mathcal{S}$, $\mathcal{T}$ with universes S, T and designated n-tuples $\overline{s}$, $\overline{t}$ of elements from S, T, respectively, are called *m-equivalent* (written $(\mathcal{S}, \overline{s}) \equiv_m (\mathcal{T}, \overline{t})$) if

$$(\mathcal{S}, \overline{s}) \models \varphi(\overline{x}) \iff (\mathcal{T}, \overline{t}) \models \varphi(\overline{x})$$

for all Sig-formulas $\varphi(\overline{x})$ of quantifier-depth $\leq m$. For the case of empty sequences $\overline{s}$ and $\overline{t}$ this means that the two structures satisfy the same sentences (formulas without free variables) of quantifier-depth at most m.

The Ehrenfeucht-Fraïssé game (short: EF-game) allows to verify the claim $(\mathcal{S}, \overline{s}) \equiv_m (\mathcal{T}, \overline{t})$. As a preparation we need the notion of partial isomorphism. Given Sig-structures $\mathcal{S}$ and $\mathcal{T}$ with universes S and T, we indicate a finite relation $\{(s_1, t_1), \ldots, (s_n, t_n)\} \subseteq S \times T$ by $\overline{s} \mapsto \overline{t}$. Such a relation is called a *partial isomorphism* if the assignment $s_i \mapsto t_i$ determines an injective (partial) function p from S to T (whose domain consists of the elements in $\overline{s}$), which moreover preserves all relations Q^S, R^S under consideration, in the sense that

$$s \in Q^S \iff p(s) \in Q^T \quad \text{and} \quad (s, s') \in R^S \iff (p(s), p(s')) \in R^T$$

for all symbols Q, R from Sig and all s, s' in the domain of p.

Let us now describe how to play the EF-game. The game $G_m((\mathcal{S}, \overline{s}), (\mathcal{T}, \overline{t}))$ is played between two players called Spoiler and Duplicator (as suggested in [FSV95]) on the structures $(\mathcal{S}, \overline{s})$ and $(\mathcal{T}, \overline{t})$. There are m rounds carried out as follows: The initial configuration is the relation $\overline{s} \mapsto \overline{t}$. Given a configuration r, a round is composed of two moves: first Spoiler picks an element s from S or t from T, and then Duplicator reacts by choosing an element in the other structure, i.e. by choosing some t from T, resp. some s from S. The new configuration is $r \cup \{(s, t)\}$. After m rounds, Duplicator has won if the final configuration is a partial isomorphism (otherwise Spoiler has won). Note that this can happen only if each intermediate configuration is also partial isomorphism. While Duplicator aims at a partial isomorphism at the end, Spoiler tries to avoid this. We say that Duplicator *wins* the game $G_m((\mathcal{S}, \overline{s}), (\mathcal{T}, \overline{t}))$ if Duplicator has a strategy to win each play (we skip a formal definition of "strategy").

Example 4.1 Let $u = aabaacaa$ and $v = aacaabaa$ and consider the game $G_2(\underline{u}, \underline{v})$ over the word models for u, v (including the linear ordering $<$ of letter positions). Duplicator looses this game: Spoiler can pick the u-positions with the letters b and c, whence Duplicator can only respond by picking the positions with b and c in v, in order to preserve the relations Q_b and Q_c; but then the order between the positions is not preserved and the constructed correspondence is not a partial isomorphism.

Example 4.2 Consider the same game as before, however over word models in the signature with successor relation S only (besides the letter predicates Q_a, Q_b, Q_c). Now Duplicator has a winning strategy: If Spoiler picks a position with b or c or a position adjacent to one of them, Duplicator reacts accordingly in the other word; in the remaining cases, where Spoiler picks the first or last position, Duplicator does the same in the other word. It is easy to check that for the second move, Duplicator will be able to respond in building a partial isomorphism, respecting the letter predicates and the successor relation. Thus Duplicator wins this game.

Example 4.3 Finally, we consider word models as labelled linear orderings (without successor relation); so we identify a word w with a structure $(\mathrm{dom}(w), <, (Q_a)_{a \in A})$. With this format of word models (and assuming the trivial alphabet $A = \{a\}$), Duplicator wins the game $G_2(aaa, a^n)$ for any $n \geq 3$: In the first round, Spoiler may pick a first position, a last position, or a non-border position in one of the two words, and Duplicator reacts accordingly. This allows Duplicator also to respond correctly (i.e., order-preserving) in the second round. Let us now consider a 3-rounds game $G_3(a^i, a^j)$: Here after the first round decompositions of the two words in the form $a^i = a^{i_1} a a^{i_2}$ and $a^j = a^{j_1} a a^{j_2}$ are reached (the displayed letters a representing the positions chosen in the first round). Remembering the 2-rounds game, Duplicator will win if i_1, j_1 are both ≥ 3 or else $i_1 = j_1$, and similarly for i_2, j_2. Clearly Duplicator can reach such a decomposition in the first round if i, j are both ≥ 7, or if $i = j$. In general, with k rounds ahead, Duplicator needs to ensure that corresponding letter-blocks delimited by chosen positions are of length $\geq 2^k - 1$ or are of the same length. In this way one sees that Duplicator wins $G_m(a^i, a^j)$ for any $i, j \geq 2^m - 1$; and by a slightly generalized argument one verifies that Duplicator also wins $G_m(w^i, w^j)$ for any word w and $i, j \geq 2^m - 1$.

How can one verify in general that Duplicator wins the game $G_m((\mathcal{S}, \overline{s}), (\mathcal{T}, \overline{t}))$? A simple approach is to specify, for each $k = 0, \ldots, m$, a set I_k of partial isomorphisms (describing configurations) which would Duplicator allow to win with k rounds ahead. Of course, $\overline{s} \mapsto \overline{t}$ should belong to I_m, all partial isomorphisms in the sets I_k should extend $\overline{s} \mapsto \overline{t}$, and any way to continue a play from a configuration in I_k should lead to a configuration in I_{k-1}. More precisely, there should be nonempty sets $I_m, \ldots, I_0$ of partial isomorphisms, each of them extending $\overline{s} \mapsto \overline{t}$, such that for all $k = m, \ldots, 1$ the following properties hold:

- *(back property)* $\forall p \in I_k\ \forall t \in T\ \exists s \in S$ such that $p \cup \{(s,t)\} \in I_{k-1}$
- *(forth property)* $\forall p \in I_k\ \forall s \in S\ \exists t \in T$ such that $p \cup \{(s,t)\} \in I_{k-1}$.

If a sequence $I_m, \ldots, I_0$ with these properties exists, we write $(\mathcal{S}, \overline{s}) \cong_m (\mathcal{T}, \overline{t})$. By induction on m one verifies that this condition holds iff Duplicator wins $G_m((\mathcal{S}, \overline{s}), (\mathcal{T}, \overline{t}))$.

Fraïssé showed in the fifties that the relations $\equiv_m$ and $\cong_m$ coincide on relational structures of finite signature; later Ehrenfeucht introduced the game theoretical formulation of $\cong_m$:

Theorem 4.4 (Ehrenfeucht-Fraïssé Theorem)
For $m \geq 0$:
$(\mathcal{S}, \overline{s}) \equiv_m (\mathcal{T}, \overline{t})$ *iff* $(\mathcal{S}, \overline{s}) \cong_m (\mathcal{T}, \overline{t})$ *iff Duplicator wins* $G_m((\mathcal{S}, \overline{s}), (\mathcal{T}, \overline{t}))$.

Proof. The second equivalence was explained above. The step from $\cong_m$-equivalence to $\equiv_m$-equivalence is easy by induction on m. For the converse, it is sufficient to describe any $\cong_m$-class by a formula of quantifier-depth m. More precisely: For each structure $(\mathcal{S}, \overline{s})$ and any given $m \geq 0$, there is a formula $\varphi^m_{(\mathcal{S},\overline{s})}(\overline{x})$ of quantifier-depth m which holds in precisely those structures $(\mathcal{T}, \overline{t})$ that are $\cong_m$-equivalent to $(\mathcal{S}, \overline{s})$.

The definition proceeds by induction on m, giving the formalization of $\cong_0$-equivalence (partial isomorphism) and of the two extension properties (back and forth):

$$\begin{aligned}
\varphi^0_{(\mathcal{S},\overline{s})}(\overline{x}) &:= \bigwedge_{\varphi(\overline{x})\ \text{atomic},\ (\mathcal{S},\overline{s}) \models \varphi(\overline{x})} \varphi(\overline{x}) \ \wedge \bigwedge_{\varphi(\overline{x})\ \text{atomic},\ (\mathcal{S},\overline{s}) \models \neg\varphi(\overline{x})} \neg\varphi(\overline{x}) \\
\varphi^{m+1}_{(\mathcal{S},\overline{s})}(\overline{x}) &:= \bigwedge_{s \in S} \exists x_{n+1} \varphi^m_{(\mathcal{S},\overline{s},s)}(\overline{x}, x_{n+1}) \ \wedge\ \forall x_{n+1} \bigvee_{s \in S} \varphi^m_{(\mathcal{S},\overline{s},s)}(\overline{x}, x_{n+1})
\end{aligned}$$

To justify this definition in case the structure $\mathcal{S}$ is infinite, one has to observe that (due to the finite signature) there are only finitely many atomic formulas involving variables from $x_1, \ldots, x_n$, and that (as verified by induction on m) the number of logically nonequivalent formulas $\varphi^m_{(\mathcal{S},\overline{s})}(\overline{x})$ is finite (for any given length of tuples $\overline{s}$). Thus the disjunction and the conjunction (over $s \in S$) in the definition of $\varphi^{m+1}_{(\mathcal{S},\overline{s})}(\overline{x})$ both range only over finitely many formulas $\varphi^m_{(\mathcal{S},\overline{s},s)}(\overline{x}, x_{n+1})$ and thus specify first-order formulas. $\square$

Let us reconsider the examples above. The Ehrenfeucht-Fraïssé Theorem says that Duplicator wins the m-round game on two word models iff they cannot be distinguished by sentences of quantifier-depth m. Recalling the first example, concerning $u = aabaacaa$ and $v = aacaabaa$ where Spoiler wins $G_2(\underline{u}, \underline{v})$, we see that there is indeed a sentence of quantifier-depth 2 in the signature with $<$ which distinguishes between u and v (namely, $\exists x \exists y (Q_b(x) \wedge x < y \wedge Q_c(y))$). On the other hand, as seen from the second example together with Theorem 4.4, no sentence of quantifier-depth 2 in which the successor relation is the only numerical relation can distinguish between u and v.

Coming back to the Example 4.3, we conclude the following:

Proposition 4.5 *The language $\{a^n \mid n \text{ is even}\}$ is not first-order definable.*

Proof. Supposing that a defining first-order sentence exists, we can eliminate the use of the successor relation S in terms of $<$, and obtain a sentence φ with $<$ only, say of quantifier-depth m. This sentence φ is satisfied in a^{2^m}. By Example 4.3 and Theorem 4.4, we have $a^{2^m} \equiv_m a^{2^m+1}$; hence φ is also satisfied in the word a^{2^m+1} (of odd length), which gives a contradiction. $\square$

In general, any first-order definable language L shares the following strong pumping property: If m is sufficiently large, then for any three words u, v, w over the alphabet under consideration we have $uv^m w \in L$ iff $uv^{m+1}w \in L$. In algebraic terms, this means that the syntactic monoid of a first-order definable language is aperiodic.

4.2 Locally Threshold Testable Sets

In this section we determine the expressive power of first-order logic over "successor structures", more generally over graphs of bounded degree. A graph with edge relation E is of degree $\leq d$ if for any vertex s there are at most d vertices t with $(s,t) \in E$ or $(t,s) \in E$. Special cases are the (binary) tree models or word models with successor relation only. Using EF-games, we show that first-order logic over graphs of bounded degree is of rather limited power; it can only express statements saying which local neighbourhoods of vertices appear in a graph and which not, and how often (counted up to some fixed threshold) such a neighbourhood may occur.

To specify neighbourhoods, we say that for a graph $\mathcal{S}$, $s \in S$, and $r \in \mathbb{N}$, the "sphere with radius r around s in $\mathcal{S}$" is the induced subgraph of $\mathcal{S}$ with vertices of distance $\leq r$ from s. (Here we assume that edges may be traversed in both directions.) This subgraph with designated center s is denoted $r{-}\mathrm{sph}(\mathcal{S}, s)$. Since we consider graphs of degree $\leq d$, there is, for any $r > 0$, only a finite number of possible isomorphism types of r-spheres. For an isomorphism type σ of r-spheres, let $\mathrm{occ}(\sigma, \mathcal{S})$ be the number of occurrences of spheres of type σ in $\mathcal{S}$. We show that any first-order formula is equivalent (over graphs of degree $\leq d$) to a statement on these occurrence numbers for finitely many types σ. Moreover, for any given formula the values $\mathrm{occ}(\sigma, \mathcal{S})$ are relevant only up to a certain threshold $q \in \mathbb{N}$.

Definition 4.6 Let $\mathcal{S} \sim_{r,q} \mathcal{T}$ if for any isomorphism type σ of spheres of radius r the numbers $\mathrm{occ}(\sigma, \mathcal{S})$ and $\mathrm{occ}(\sigma, \mathcal{T})$ are either both larger than the threshold number q or else coincide. A set of graphs is *locally threshold testable* if for some r, q, it is a union of $\sim_{r,q}$-equivalence classes (which is a finite union if the degrees of the graphs under consideration are bounded).

The following theorem states that $\sim_{r,q}$-equivalence (for suitable r, q) is fine enough to capture m-equivalence (i.e., indistinguishability by formulas of quantifier depth m). More general formulations are possible, but for simplicity we stay with the case of graphs of degree bounded by d.

Theorem 4.7 ("Sphere Theorem", [Hnf65])
For any $m \geq 0$ there are $r, q \geq 0$ such that for any two graphs $\mathcal{S}, \mathcal{T}$ (finite or infinite, but of degree $\leq d$) we have: If $\mathcal{S} \sim_{r,q} \mathcal{T}$ then $\mathcal{S} \equiv_m \mathcal{T}$.

Proof. By Theorem 4.4, it suffices to ensure $\mathcal{S} \cong_m \mathcal{T}$ from $\mathcal{S} \sim_{r,q} \mathcal{T}$ for suitably chosen r, q. Set $r = 3^m$ and $q = m \cdot c$ where c is the maximal size of a 3^m-sphere. The required sequence of sets $I_0, \ldots, I_m$ of partial isomorphisms is defined as follows: Let $p : (s_1, \ldots, s_{m-k}) \mapsto (t_1, \ldots, t_{m-k})$ belong to I_k iff

$$\bigcup_{i=1}^{m-k} 3^k - \mathrm{sph}(\mathcal{S}, s_i) \quad \cong \quad \bigcup_{i=1}^{m-k} 3^k - \mathrm{sph}(\mathcal{T}, t_i)$$

i.e., the two induced subgraphs formed from the 3^k-spheres around the s_i, resp. the t_i, are isomorphic. To verify e.g. the forth property, assume this condition holds for p and let $s (= s_{m-(k-1)}) \in S$. We have to find $t (= t_{m-(k-1)}) \in T$ such that

$$\bigcup_{i=1}^{m-(k-1)} 3^{k-1} - \mathrm{sph}(\mathcal{S}, s_i) \quad \cong \quad \bigcup_{i=1}^{m-(k-1)} 3^{k-1} - \mathrm{sph}(\mathcal{T}, t_i).$$

If $s \in \frac{2}{3} \cdot 3^k - \mathrm{sph}(\mathcal{S}, s_i)$ for some s_i, we may choose t from $\frac{2}{3} \cdot 3^k - \mathrm{sph}(\mathcal{T}, t_i)$ correspondingly; note that $3^{k-1} - \mathrm{sph}(\mathcal{S}, s)$ is contained in $3^k - \mathrm{sph}(\mathcal{S}, s_i)$ and thus $3^{k-1} - \mathrm{sph}(\mathcal{T}, t)$ in $3^k - \mathrm{sph}(\mathcal{T}, t_i)$. So $3^{k-1} - \mathrm{sph}(\mathcal{S}, s) \cong 3^{k-1} - \mathrm{sph}(\mathcal{T}, t)$ holds. Otherwise, $3^{k-1} - \mathrm{sph}(\mathcal{S}, s)$, say of type σ, is disjoint from all $3^{k-1} - \mathrm{sph}(\mathcal{S}, s_i)$, and it suffices to find a sphere of type σ in $\mathcal{T}$ which is disjoint from all spheres $3^{k-1} - \mathrm{sph}(\mathcal{T}, t_i)$. This will be possible if the number of occurrences of spheres of type σ in $\mathcal{T}$ is large enough. But this is guaranteed in view of the number of these spheres in $\mathcal{S}$ and the assumption $\mathcal{S} \sim_{r,q} \mathcal{T}$. $\square$

As a consequence, we note the following result:

Theorem 4.8 *A first-order definable set of graphs of bounded degree is locally threshold testable.*

Thus, first-order logic over words, trees, and graphs of bounded degree, can express only "local properties", i.e. statements on occurrences of local neighbourhoods. More precisely, each first-order formula is equivalent to a boolean combination of statements "sphere σ occurs $\geq n$ times" (because for σ of radius r and $n \leq q$, such conditions fix the $\sim_{r,q}$-class to which a structure belongs). The statement "sphere σ (of k elements) occurs $\geq n$ times" can be expressed by a sentence of the form

$$\exists x_1 \exists \overline{y_1} \ldots \exists x_n \exists \overline{y_n} \varphi(x_1, \overline{y_1}, \ldots, x_n, \overline{y_n})$$

where each $\overline{y_i}$ is a $(k-1)$-tuple of variables and the formula φ states the following: The x_i are pairwise distinct (as centers of spheres), for each i the elements x_i and $\overline{y_i}$ are pairwise distinct building a graph of isomorphism type σ (expressed by a conjunction of all atomic formulas and their negations which hold over the elements x_i, $\overline{y_i}$ to form a sphere of type σ), and for any element z distinct from x_i and the $\overline{y_i}$ the distance from x_i is greater than d (i.e. is not adjacent to one of those elements of $\overline{y_i}$ which were chosen in distance $< d$ to x_i). When written in prenex normal form, this sentence is of Σ_2-form.

Corollary 4.9 *Over graphs of bounded degree, any first-order sentence is equivalent to a boolean combination of Σ_2-sentences.*

This strong reduction of quantifier complexity of formulas shows again the weakness of first-order logic over graphs.

In the domain of words, we see that a language is FO[S]-definable iff it is locally threshold testable ([Th82a]). The locally threshold testable word languages are usually introduced in a slightly different but equivalent way than above. Note that prefixes and suffixes of words may be specified by spheres whose center has no predecessor, respectively has no successor. When spheres are replaced just by (word-) segments without designated center (as is usually done in language theory), prefixes and suffixes have to be treated separately: For a word w over an alphabet A, a word $\sigma \in A^+$, and a number q let $\mathrm{occ}_{\sigma,q}(w)$ be the number of occurrences of σ in w if this number is $< q$, and otherwise q. Furthermore, let $\mathrm{pref}_d(w)$ and $\mathrm{suf}_d(w)$ be the segment of the first, resp. last d letters of w (or w itself if $|w| < d$). Now define, for words u, v and given d and q, $u \sim_{d,q} v$ if for all segments σ of length $\leq d$ we have $\mathrm{occ}_{\sigma,q}(u) = \mathrm{occ}_{\sigma,q}(v)$, $\mathrm{pref}_d(u) = \mathrm{pref}_d(v)$, and $\mathrm{suf}_d(u) = \mathrm{suf}_d(v)$. A language is then called *locally threshold testable* if it is a union of $\sim_{d,q}$-equivalence classes for some d and q.

As an example, consider the language L defined by the regular expression $a^*ba^*ca^*$. For any d and q, one finds a number n such that $a^n ba^n ca^n \sim_{d,q} a^n ca^n ba^n$ (in accordance with Example 4.2 given above for EF-games). Since the first word belongs to L and the second does not, L is not locally threshold testable, hence not FO[S]-definable.

Theorem 4.8 can also be applied to obtain linear time bounds for first-order expressible graph problems ([See96]). Furthermore, it yields a new proof for the reduction of EMSO-logic to finite automata: An EMSO-formula defines a projection of a first-order definable set and hence, by Theorem 4.8, a projection of a locally threshold testable set. Since locally threshold testable languages are recognized by finite automata, so are their projections (using nondeterminism). In [Th91] this approach is considered over graphs of bounded degree and taken as a starting point to introduce finite-state graph acceptors. In the framework of pictures (rectangular arrays of symbols, with a horizontal and a vertical successor relation), these graph acceptors turn out to be equivalent to "tiling systems" (cf. [GRST96] or [GR96]).

4.3 Star-Free Languages

A language $L \subseteq \Sigma^+$ is called *star-free* if it can be constructed from finite languages by applications of boolean operations and concatenation. Accordingly, star-free expressions over a given alphabet A are built up from constants $\emptyset$, ϵ and $a \in A$ (denoting the empty set, the singleton with the empty word, and the set $\{a\}$, respectively) by means of the operations $\cup$, $\cap$, $\sim$ (for complement w.r.t. A^*), and concatenation dot $\cdot$. The expression A^* is also admitted, as abbreviation of $\sim \emptyset$. By a natural correspondence between these operations and the logical connectives $\vee$, $\wedge$, $\neg$, and $\exists$, it is easy to transform star-free expressions into first-order formulas. For example, over $A = \{a, b, c\}$ the expression $A^* \cdot a \cdot b \cdot \sim (A^* \cdot a \cdot A^*)$ defines the same language as

$$\exists x \exists y (S(x,y) \wedge Q_a(x) \wedge Q_b(y) \wedge \neg \exists z (y < z \wedge Q_a(z))).$$

An analysis of first-order definability shows also the converse:

Theorem 4.10 (McNaughton, Papert [McNP71])
A language is star-free iff it is first-order definable (in the signature with $<$).

Proof. For the translation of first-order formulas into star-free expressions, we follow the approach of [Lad77], [Th82a], [PP86], applying the Ehrenfeucht-Fraïssé technique.

We proceed by induction on quantifier-depth m and sketch the induction step. Consider a first-order formula $\varphi(x_1, \ldots, x_n)$ of quantifier-depth m, where for simplicity we assume that φ implies $x_1 < \ldots < x_n$. We shall reformulate $\varphi(x_1, \ldots, x_n)$ as a disjunction of "normal formulas"

$$\psi_0 \wedge Q_{a_1}(x_1) \wedge \psi_1 \wedge \ldots \wedge Q_{a_n}(x_n) \wedge \psi_n$$

where the ψ_i, again of quantifier-depth m, speak only about the segments enclosed by the x_i; i.e. ψ_0 speaks only about the segment up to (and excluding) x_1, for $0 < i < n$ the formula ψ_i speaks only about the segment enclosed by x_i and x_{i+1}, and ψ_n speaks about the segment after x_n to the end of the word. (Formally, this is ensured by allowing only relativized quantifiers in the ψ_i, e.g. in ψ_1 only quantifiers $\exists y(x_1 < y < x_2 \wedge \dots)$ and $\forall y(x_1 < y < x_2 \rightarrow \dots)$.) Given the reduction to disjunctions of normal formulas, the induction is easy: in the induction step, a formula such as $\exists x\varphi(x)$ is written as a disjunction of normal formulas $\exists x(\psi_0 \wedge Q_a(x) \wedge \psi_1)$, where for ψ_0, ψ_1 equivalent star-free expressions r_0, r_1 exist by induction hypothesis. Then $\exists x\varphi(x)$ is equivalent to the corresponding union of expressions $r_0 \cdot a \cdot r_1$.

To achieve the reduction of formulas to disjunctions of normal formulas, two facts are used on the m-equivalence between word models, which can be verified with the Ehrenfeucht-Fraïssé game technique: first, the description of the $\equiv_m$-class of a word model $(\underline{w}, p_1, \dots, p_n)$ with designated positions $p_1, \dots, p_n$ by a formula $\varphi^m_{(\underline{w},p_1,\dots,p_n)}$ (cf. the proof of Theorem 4.4), and secondly the following "congruence lemma" for m-equivalence:

Lemma 4.11 *If $\underline{u} \equiv_m \underline{u'}$, $a \in A$, and $\underline{v} \equiv_m \underline{v'}$, then $\underline{u \cdot a \cdot v} \equiv_m \underline{u' \cdot a \cdot v'}$.*

Proof. We use the Ehrenfeucht-Fraïssé Theorem 4.4. The assumption tells us that Duplicator has winning strategies for the games $G_m(\underline{u}, \underline{u'})$ and $G_m(\underline{v}, \underline{v'})$. An obvious composition of these two strategies ("on the segments u and u' use the first strategy, on the segments v and v' use the second strategy") guarantees Duplicator to win also the game $G_m(\underline{u \cdot a \cdot v}, \underline{u' \cdot a \cdot v'})$. □

Now a first-order formula $\varphi(x_1, \dots, x_n)$ of quantifier-depth m (assuming $x_1 < \dots < x_n$ as before) is transformed into a normal formula as follows. By definition of m-equivalence $\equiv_m$, the class of word models $(\underline{v}, q_1, \dots, q_n)$ which satisfy φ is a (finite) union of $\equiv_m$-classes, each of them described by a formula $\varphi^m_{(\underline{w},p_1,\dots,p_n)}(x_1, \dots, x_n)$ where $(\underline{w}, p_1, \dots, p_n)$ is a representative of the respective class. So φ is equivalent to a disjunction of formulas $\varphi^m_{(\underline{w},p_1,\dots,p_n)}(x_1, \dots, x_n)$, and it suffices to express such a formula as a disjunction of normal formulas. By the lemma above the $\equiv_m$-class of $(\underline{w}, p_1, \dots, p_n)$ is fixed by the $\equiv_m$-classes of the segments $w_0, \dots, w_n$ of w delimited by the positions p_i, and by the letters a_i associated with the p_i. We now collect conjunctions of corresponding formulas $\varphi^m_{w_i}$ and $Q_{a_i}(x_i)$ (each conjunction describing a sequence $w_0, a_1, \dots, a_n, w_n$), namely those conjunctions which imply $\varphi^m_{(\underline{w},p_1,\dots,p_n)}(x_1, \dots, x_n)$. Altogether we obtain a formula equivalent to φ, as a disjunction of conjunctions of formulas $\varphi^m_{w_i}$ and $Q_{a_i}(x_i)$. Thus a disjunction of normal formulas is reached. □

Theorem 4.10 is the starting point of a rich definability theory of star-free languages. The significance of the class of star-free languages is much supported by Schützenberger's fundamental characterization [Sch65] in terms of finite group-free (or: aperiodic) monoids. (Whereas we have verified the aperiodicity property of first-order definable languages in Proposition 4.5 above, the converse is more difficult and relies on nontrivial results concerning the decomposition of monoids; see e.g. [Per90].) In the framework of minimal deterministic automata, this aperiodicity property amounts to the lack of words which induce a nontrivial permutation on a subset of the state space. Since this property is decidable, or equivalently whether the syntactic monoid of a regular language contains a nontrivial group, Schützenberger's theorem together with Theorem 4.10 provides an algorithm to decide whether a regular language is first-order definable. Many more language classes have been characterized by special types of regular expressions, restrictions and variants of first-order formulas, structural properties of automata, and corresponding properties of syntactic monoids. The class of locally threshold testable languages, considered in the previous section, is an example, and adaptations of the Ehrenfeucht-Fraïssé method are usually applied in fixing the logical part of the characterizations. The field is presented in several surveys and books, e.g. [Pin86] and [Str94]. Below we list only a small selection of results. We shall only consider languages of words; in fact, it seems difficult to characterize first-order logic over trees by structural properties of tree automata (for partial results in this direction see [Pot95]).

Within the class of star-free languages, a hierarchy $(\mathcal{V}_n)_{n\geq 0}$ of language classes can be built up whose levels measure the concatenation depth of defining star-free expressions. Fixing an alphabet A with at least two letters, one sets $\mathcal{V}_0$ to be the class consisting of the languages $\emptyset$ and A^*, and $\mathcal{V}_{n+1}$ to be the class of boolean combinations of languages $L_0 \cdot a_1 \cdot L_1 \cdot \ldots a_k \cdot L_k$ with $k \geq 0$, $a_i \in A$, and $L_i \in \mathcal{V}_n$. The levels of this hierarchy have a natural characterization in terms of quantifier-prefixes of first-order formulas in which only $<$ (but not S) appears as a numerical relation:

Theorem 4.12 (cf. [PP86]) *A star-free language belongs to the class $\mathcal{V}_n$ iff it is definable by a $B(\Sigma_n)$-sentence of first-order logic with $<$.*

The first level of this hierarchy gives the class of piecewise testable languages (Simon [Sim75]). It can be shown that the $\mathcal{V}_n$ form a strictly increasing hierarchy; in the logical framework the EF-game may be applied for the hierarchy proof ([Th84a]). Analogous results can be proved for the closely related "dot-depth hierarchy" ([Th82a], [Th87]). A still finer classification, distinguishing formulas not only by the number of quantifier alternations but also by the lengths of quantifier blocks, is studied in [BS95]. An open problem in this theory is the question whether the smallest n such that a given star-free language belongs to $\mathcal{V}_n$ can be computed effectively.

The most elementary examples of languages which are not first-order definable are based on "modular counting"; instances are the set of words of even length or the language PARITY over $\{a, b\}$ consisting of the words with an even number of occurrences of b (cf. Proposition 4.5 above). Two kinds of extensions of first-order logic have been considered to obtain stronger frameworks within the expressive range of MSO-logic where such languages become definable: the adjunction of stronger quantifiers (or similar operators), and the use of more general numerical relations than successor and order.

Properties which involve modular counting are conveniently described by *modular quantifiers* $\exists^{q,r}x$, to be read as "there are exactly r elements x modulo q such that ...". In [STT95] it is shown that the languages definable in the extension of first-order logic by modular quantifiers are those whose syntactic monoid is finite and contains only groups which are solvable. As a consequence, this class is properly included in the class of regular languages, and membership of a regular language in it is decidable.

Certain properties concerning modular counting are also captured by special numerical predicates, in particular the unary predicates $C_{r,q}$, containing those numbers (in word models: positions) which are congruent to r modulo q. The use of these predicates was suggested in [McNP71], and together with successor and order they constitute the *regular numerical predicates*. Using them, the set of words of even length becomes definable, whereas PARITY is not definable in first-order logic with regular numerical predicates only.

Such extensions of FO$[S,<]$-logic by additional numerical relations are closely connected with circuit complexity classes. Let us mention two fundamental theorems, which are the entrance to a fascinating and fastly developing theory: A language is definable in first-order logic with arbitrary numerical relations iff it belongs to the circuit complexity class AC^0, i.e. is defined by a family of circuits of bounded depth and with "and"-gates and "or"-gates of unbounded fan-in ([Imm87]). As shown in [BCST88], the intersection of AC^0 with the class of regular languages contains precisely the languages definable in first-order logic with regular numerical predicates. The reader should consult Straubing's book [Str94] for proofs, many more results, and some intriguing open problems.

In this section we discussed three typical applications of the Ehrenfeucht-Fraïssé technique to first-order definable formal languages: the confinement of FO$[S]$-logic to the definition of local properties only (Theorems 4.7 and 4.8), the inability of FO$[S,<]$-logic to specify conditions on modular counting (Proposition 4.3), and the compatibility of FO$[S,<]$-definability with concatenation (Congruence Lemma 4.11). Another important application of model theoretic games in the theory of automata and transition systems, which we cannot discuss further here, is the notion of *bisimulation*. Bisimulations can be viewed as special families of partial isomorphisms, corresponding to a restricted type of EF-game. This game is played on tree structures arising from unravellings of transition systems, and a play of the game is required to proceed only "downward" the two trees under consideration, starting from the roots. The corresponding logics are systems of modal logic and process logic (equivalent to fragments of first-order logic). For an overview of this subject see [Mil90] or [Sti96].

5 Automata and MSO-Logic on Infinite Words

In his paper [Bü62], Büchi showed that MSO-logic over ω-words is equivalent to finite automata equipped with natural acceptance conditions for infinite words. This founded a beautiful branch of definability theory for properties of infinite sequences, complementing earlier results of descriptive set theory and recursion theory.

In this section, only some central logical aspects of ω-automata are reviewed. More information can be found in the chapter on ω-languages in this Handbook or the surveys [HR86], [Sta87], [Th90], [TL94].

5.1 ω-Automata

While the acceptance condition of automata over finite words is rather canonical, there are many possibilities of defining acceptance of infinite words. An acceptance condition restricts the occurrences of states in a run ρ under consideration. Usually, it refers to those states which occur infinitely often in ρ and is fixed by an "acceptance component" of the ω-automaton.

Definition 5.1 A *finite ω-automaton* has the form $\mathcal{A} = (Q, A, q_0, \Delta, Acc)$ with finite state set Q, input alphabet A, initial state q_0, transition relation $\Delta \subseteq Q \times A \times Q$, and an acceptance component Acc. A run of $\mathcal{A}$ on a given input ω-word $\alpha = \alpha(0)\alpha(1)\ldots$ with $\alpha(i) \in A$ is a sequence $\rho = \rho(0)\rho(1)\ldots \in Q^\omega$ such that $\rho(0) = q_0$ and $(\rho(i), \alpha(i), \rho(i+1)) \in \Delta$ for $i \geq 0$. In deterministic automata the transition relation is replaced by a transition function $\delta : Q \times A \to Q$, and a run has to satisfy $\rho(i+1) = \delta(\rho(i), \alpha(i))$ for $i \geq 0$.

Let us introduce the standard acceptance modes. We write $\exists^\omega$ for the quantifier "there exist infinitely many" and consider the set

$$\mathrm{In}(\rho) = \{q \in Q \mid \exists^\omega i\ \rho(i) = q\}.$$

The most frequently used acceptance conditions are the following requirements on $\mathrm{In}(\rho)$ (called so according to their inventors):

- *Büchi condition* [Bü62]: $\mathrm{In}(\rho) \cap F \neq \emptyset$ for a set $F \subseteq Q$ of "final states", requiring that some final state occurs infinitely often in the run ρ.

- *Muller condition* [Mul63]: $\bigvee_{F \in \mathcal{F}} \mathrm{In}(\rho) = F$, for a family $\mathcal{F} \subseteq 2^Q$ of final state sets, requiring that the set of states assumed infinitely often in the run ρ forms a set in $\mathcal{F}$.

- *Rabin condition* ("pairs condition") [Rab69], [Rab72]: $\bigvee_{i=1}^{n}(\mathrm{In}(\rho) \cap E_i = \emptyset\ \wedge\ \mathrm{In}(\rho) \cap F_i \neq \emptyset)$, for a sequence Ω of "accepting pairs" $(E_1, F_1), \ldots, (E_n, F_n)$ with $E_i, F_i \subseteq Q$; it requires that for some i, all states of E_i are visited only finitely often in ρ (<u>e</u>xcluded from some point onwards), but some state of F_i (i.e., a <u>f</u>inal state) is visited infinitely often.

- *Streett condition* ("complemented pairs condition", the dual of the Rabin condition) [St82]: $\bigwedge_{i=1}^{n}(\mathrm{In}(\rho)\cap E_i \neq \emptyset \vee \mathrm{In}(\rho)\cap F_i = \emptyset)$ for a sequence Ω of pairs $(E_1, F_1), \ldots, (E_n, F_n)$ where the E_i, F_i are subsets of Q; it represents a "fairness condition" which can be read as "for each i, if some state of F_i is visited infinitely often, then some state of E_i is visited infinitely often."

Thus, an ω-automaton $\mathcal{A} = (Q, A, q_0, \Delta, F)$ (used with the Büchi acceptance condition) is called *Büchi automaton*; similarly, we speak of *Muller automata* $\mathcal{A} = (Q, A, q_0, \Delta, \mathcal{F})$, *Rabin automata*, and *Streett automata* $\mathcal{A} = (Q, A, q_0, \Delta, \Omega)$, respectively, according to the use of their acceptance component. An ω-language is called *Büchi-, Muller-, Rabin-, Streett-recognizable* if it consists of the ω-words (over the considered alphabet) which are accepted by a Büchi-, Muller-, Rabin-, Streett-automaton, respectively.

It is useful to compare these acceptance conditions in a simple case.

Example 5.2 Consider the ω-language $L \subseteq \{a, b, c\}^\omega$ consisting of all ω-words α which satisfy the condition "if a occurs infinitely often in α, then also b does". The most convenient option is to use a Streett automaton for defining L, which has three states q_a, q_b, q_c visited after reading a, b, c, respectively. The acceptance component Ω just contains the pair $(\{q_b\}, \{q_a\})$. A corresponding Muller automaton (over the same state graph) has the acceptance component $\mathcal{F}$ which contains all sets F for which the implication $q_a \in F \Rightarrow q_b \in F$ holds. For obtaining a suitable Rabin automaton, note that α is in L iff either b occurs infinitely often in α or both a, b occur only finitely often. Thus, two accepting pairs suffice (again over the same state graph), namely $(\emptyset, \{q_b\})$ and $(\{q_a, q_b\}, \{q_c\})$. To obtain a suitable Büchi automaton, we capture the disjunction "either b infinitely often or a, b finitely often" by nondeterminism: we introduce an extra "c-sink-state" $q_{c!}$, reached via letter c from any of q_a, q_b, q_c and such that only one transition from $q_{c!}$ exists, via c back to $q_{c!}$. Now we may set $F = \{q_b, q_{c!}\}$ as set of final states.

An exercise in simulation shows that the above example is typical:

Proposition 5.3 *Nondeterministic Büchi-, Muller-, Rabin-, and Streett-automata all recognize the same class of ω-languages.*

Proof. A Büchi-, Rabin-, or Streett-automaton is easily simulated by a Muller automaton, by collecting, in its acceptance component $\mathcal{F}$, those state sets which lead to acceptance in the given automaton. In turn, given a Muller automaton with acceptance component $\mathcal{F}$, a corresponding Büchi automaton guesses in advance the set $F \in \mathcal{F}$ of states to be visited infinitely often, and also guesses the point on its input α from which onwards only states in F will be seen. From there it suffices to check that the visited states fill the set F again and again. This can be signalled by the Büchi acceptance condition. $\square$

It follows from McNaughton's Theorem (see next section) that even deterministic automata can be obtained when the Muller-, Rabin-, or Streett-acceptance is used. For the complexity analysis of the transformations from one acceptance condition to another, see e.g. [Saf88], [Saf92] and [KPB95].

In a more general framework of acceptance conditions, one can also consider the case that only the mere occurrence of states in runs is restricted (instead of the infinite occurrence). For a given run ρ one considers the set

$$\mathrm{Oc}(\rho) = \{q \in Q \mid \exists i \; \rho(i) = q\}$$

and may form analogous expressions as above, e.g. $\mathrm{Oc}(\rho) \cap F \neq \emptyset$ for a set F of states or $\mathrm{Oc}(\rho) \in \mathcal{F}$ for a system $\mathcal{F}$ of state sets. The latter is called *Staiger-Wagner acceptance* (introduced in [SW74]); it captures the general case of condition where the set of visited states in a run determines whether the input is accepted. (In the classification of execution sequence properties of nonterminating programs, this case is described by the term "obligation property", cf. [MP92].)

A still more flexible framework is obtained in a logical setting: Here we consider acceptance components which are boolean combinations of formulas "$\exists i \; \rho(i) \in F$" and "$\exists^\omega i \; \rho(i) \in F$" for state sets F of a given automaton. All conditions mentioned above can be formulated in this way. A natural classification leads to six classes, given by the mentioned "atomic conditions", their negations, and boolean combinations of the first, resp. second type of atomic formula. Boolean combinations of conditions "$\exists i \; \rho(i) \in F$" characterize the Staiger-Wagner-acceptance mode, while Muller acceptance is described by boolean combinations of conditions "$\exists^\omega i \; \rho(i) \in F$". A complete analysis of the expressiveness of the acceptance conditions and their transfer to automata with arbitrary storage types (like pushdown store) is given by Staiger [Sta87] and Engelfriet and Hoogeboom [EH93].

Let us connect the Büchi recognizable ω-languages with the standard notion of regular sets of finite words. Suppose the Büchi automaton $\mathcal{A} = (Q, A, q_0, \Delta, F)$ accepts the ω-word α, say by a run which reaches a final state $q \in F$ and revisits this final state again and again. Let U_q, V_q be the (regular) sets of words which allow $\mathcal{A}$ to pass from q_0 to q, resp. from q to q. Then the ω-word α can be decomposed as $\alpha = uv_0v_1\ldots$ with $u \in U_q$, $v_i \in V_q$ for $i \geq 0$, in short, $\alpha \in U_q \cdot V_q^\omega$. Thus we have $L_\omega(\mathcal{A}) = \bigcup_{q \in F} U_q \cdot V_q^\omega$. It is not difficult to show that this form of ω-languages characterizes Büchi recognizability: An ω-language is Büchi recognizable iff it is a finite union of sets $U \cdot V^\omega$ where U, V are regular sets of finite words. One speaks of the *regular ω-languages*.

In the sequel we focus on the Büchi recognizable (or regular) ω-languages and their logical description. The key result, due to Büchi [Bü62], states that an ω-language is Büchi recognizable iff it is MSO-definable. The nontrivial step in the proof is to show closure of the class of Büchi recognizable sets under complement. The original approach of [Bü62] uses a representation of Büchi recognizable sets in the form $\bigcup_{1 \le i \le n} U_i \cdot V_i^{\omega}$, where the U_i, V_i are classes of a sufficiently fine congruence over A^* of finite index, and applies a combinatorial argument (e.g., a form of Ramsey's Theorem) to guarantee that the complement has again such a representation.

An alternative is to proceed to deterministic automata. This approach does not work when the Büchi acceptance condition is employed. (For example, a deterministic Büchi automaton recognizes the set of ω-words over $\{a, b\}$ with infinitely many occurrences of a, but no deterministic Büchi automaton recognizes the complement of this set.) However, it turns out that deterministic Muller automata are equivalent in expressive power to (nondeterministic) Büchi automata. The complementation result follows, because the class of ω-languages recognized by deterministic Muller automata is clearly closed under complement. (In an automaton with state set Q and system $\mathcal{F}$ of final state sets, proceed to $2^Q \setminus \mathcal{F}$.) In the next two subsections we give a proof of this determinization theorem and discuss some of its logical applications.

5.2 Determinization of ω-Automata

The purpose of this section is to show the key theorem of the theory of finite ω-automata:

Theorem 5.4 (McNaughton's Theorem [McN66])
A Büchi automaton can be transformed effectively into an equivalent deterministic Muller automaton.

We shall follow Safra's proof [Saf88], which is an intricate refinement of the classical subset construction as used in the determinization of automata over finite words. First we outline the main ideas and do some preparations.

Let $\mathcal{A} = (Q, A, q_0, \Delta, F)$ be a Büchi automaton. The classical subset construction uses sets of states from Q, which we call $(Q-)$*macrostates* here, as states of the desired deterministic automaton, and such a macrostate is declared final if it contains a state from F. Starting from $\{q_0\}$, the subset automaton will assume after a finite input word w the macrostate consisting of all states reachable by $\mathcal{A}$ from q_0 via w. However, the acceptance of an ω-word by $\mathcal{A}$ cannot be captured by this construction: For instance, assume $F = \{q\}$ and that $\mathcal{A}$ can reach q via each prefix of α, but that no such run ending in q can be continued on the given input ω-word (while it is continued from other reachable states). Then "success" is signalled by each macrostate of the run of the subset automaton (since q is present in all macrostates), but no infinite run of $\mathcal{A}$ on the input exists in which q occurs infinitely often.

In Safra's construction, an own thread of macrostates is split off whenever final states are encountered. The different macrostates which are to be handled simultaneously are organized in a tree structure, called *Safra tree*. Safra trees will serve as states of the deterministic automaton to be constructed. The root macrostate of such a Safra tree collects the momentary reachable states of the given automaton $\mathcal{A}$, as in the classical subset automaton. The "splitting of threads" is realized by a simple rule: For each macrostate occurring in a given Safra tree in which final states (from F) are present, say the states $f_1, \ldots, f_m$, introduce $\{f_1, \ldots, f_m\}$ as a new son-macrostate (more precisely, as the youngest son in the order of sons). To proceed to the next Safra tree in the run, apply the usual subset construction macrostate-wise for each macrostate in the Safra tree (including the newly created son-macrostates), i.e. compute the set of states reachable from the respective macrostate via the input letter under consideration. Note that in this way the union of son-macrostates is always initialized and henceforth kept as a subset of the corresponding parent macrostate.

Without a process of merging macrostates, this construction will lead to trees of unbounded size. To obtain a finite bound on the size of Safra trees, two merge operations are performed, which we call "horizontal" and "vertical" (referring to the usual display of trees). A horizontal merge causes deletion of a state q in all macrostates for which also an older-brother macrostate with q exists. (Empty macrostates arising in this way are deleted from the Safra tree.) This makes brother macrostates disjoint, allowing at most $|Q|$ sons for a given parent. The vertical merge, which will need some extra justification, causes deletion of all sons of a macrostate (with all their descendants) if the union of these son macrostates equals the parent macrostate. When this happens, we say a "breakpoint" is reached for the parent macrostate. Due to the vertical merge, the union of brother-macrostates in Safra trees is always a *proper* subset of the associated parent macrostate; thus the height (length of a longest path) of Safra trees is bounded by $|Q| - 1$.

Let us analyze the role of breakpoints in the context of the subset construction. Assume that on a given input word, we start from some macrostate R_0, reach after reading input u_1 the macrostate Q_1 which contains a nonempty subset $F_1 \subseteq F$, and continuing the run (starting now with F_1 as son of Q_1) reach after reading v_1 a breakpoint, i.e. we reach a set R_1 from Q_1 and a set G_1 from F_1 such that $G_1 = R_1$. Clearly, in this situation, any state in R_1 is reachable by $\mathcal{A}$ from some R_0-state via $u_1 v_1$ with an intermediate visit in F (namely, in F_1). Suppose we continue in this way, as indicated in the figure.

$$
\begin{array}{ccccccccccc}
R_0 & \overset{u_1}{\rightsquigarrow} & Q_1 & \overset{v_1}{\rightsquigarrow} & R_1 & \overset{u_2}{\rightsquigarrow} & & \overset{u_i}{\rightsquigarrow} & Q_i & \overset{v_i}{\rightsquigarrow} & R_i \\
 & & \cup\!| & & \| & & \cdots & & \cup\!| & & \| \\
 & & F_1 & \overset{v_1}{\rightsquigarrow} & G_1 & & & & F_i & \overset{v_i}{\rightsquigarrow} & G_i
\end{array}
$$

By $R_i = G_i$, $F_i \overset{v_i}{\rightsquigarrow} G_i$, $F_i \subseteq Q_i$, and $R_{i-1} \overset{u_i}{\rightsquigarrow} Q_i$ one obtains (inductively on $i > 0$): *For all $q \in R_i$ there is a $p \in R_0$ such that $\mathcal{A}$ reaches from p the state q*

via $u_1v_1\dots u_iv_i$, passing i times through F, namely, at least once on each of the segments u_jv_j.

If between R_i and R_{i+1} (at breakpoints) more son macrostates (of final states) are created than just F_{i+1}, this claim holds by the same argument. Let us note an interesting consequence:

Remark 5.5 *Suppose $R_0, R_1, \dots$ is a macrostate sequence of $\mathcal{A}$, obtained by starting in $R_0 = \{q_0\}$ and applying the subset construction, such that (for $i > 0$) R_i is the i-th breakpoint macrostate, reached after input $w_1 \dots w_i$, respectively. Then there exists a successful run of $\mathcal{A}$ on the ω-word $w_1w_2\dots$.*

Proof. For $i > 0$ and any $q \in R_i$ pick an $\mathcal{A}$-run on $w_1 \dots w_i$ from q_0 to q which passes i times through F, namely at least once on each of the segments w_j (as shown above). These finite runs form a tree which is finitely branching and infinite. An application of König's Lemma yields some (infinite) run of $\mathcal{A}$ on $w_1w_2\dots$ with infinitely many visits to F. □

Thus, an infinite sequence of breakpoints can serve to detect a successful $\mathcal{A}$-run. It will be seen that this method to detect successful $\mathcal{A}$-runs is complete.

Proof of McNaughton's Theorem. Given the Büchi automaton $\mathcal{A} = (Q, A, q_0, \Delta, F)$ the desired Muller automaton $\mathcal{B} = (Q_\mathcal{B}, A, q_{0\mathcal{B}}, \delta, \mathcal{F})$ is defined as follows: Let $Q_\mathcal{B}$ be the set of all Safra trees over Q; these are ordered trees labelled by Q-macrostates, such that brother macrostates are disjoint and their union is a proper subset of the respective parent macrostate. We allow that any macrostate in a Safra tree may be marked, say by "!" (which will indicate the occurrence of breakpoints).

Formally, one distinguishes between a node of a Safra tree, which is named by a positive natural number, and its label, which is either a macrostate or a pair of a macrostate and the mark "!". Names of deleted nodes may be reused. If there are at most m nodes in the Safra trees under consideration, names from the set $\{1, \dots, 2m\}$ will be sufficient; the m extra names are used to handle the interplay between deletion and creation of nodes correctly: If in a sequence $s_1, s_2, \dots$ of successive Safra trees (on some input) a node name k appears in each tree, say labelled with macrostate $R_1, R_2, \dots$, respectively, then we shall need that a thread of states $q_1, q_2, \dots$ with $q_i \in R_i$ indeed represents an $\mathcal{A}$-run on the considered input. If node names could be reused immediately after deletion (e.g. due to horizontal merge) when a new node is to be created (due to visits of final states), the $\mathcal{A}$-runs would be confused. So we keep the names for nodes which stay, but take for any newly created node a name which does not occur in the previous Safra tree (using, if necessary, the reservoir of extra names $m+1, \dots, 2m$).

As initial state $q_{0\mathcal{B}}$ one takes the Safra tree consisting just of the root macrostate $\{q_0\}$. For a given Safra tree s and an input letter a, the value $\delta(s,a)$ of the transition function δ is determined in stages as mentioned above:

1. For any macrostate R in s with states from F add a node as youngest son, labelled with macrostate $R \cap F$,

2. apply the subset construction, i.e., replace any macrostate R by the set $\{q \in Q \mid \exists r \in R\ (r,a,q) \in \Delta\}$

3. apply the horizontal and then the vertical merge as explained above, marking a parent macrostate with "!" if all sons are deleted by the vertical merge.

Finally, let a set S of Safra trees be in the system $\mathcal{F}$ of final state sets if some node k appears in all Safra trees of S, and k is marked by "!" at least once in S.

The proof is finished by showing $L_\omega(\mathcal{A}) = L_\omega(\mathcal{B})$.

If $\mathcal{B}$ accepts the ω-word α, then, by the definition of $\mathcal{F}$, in the successful run of Safra trees of $\mathcal{B}$ some node k finally stays and is marked by "!" infinitely often. The argument of Remark 5.5 above, applied to the sequence $R_1, R_2, \ldots$ of the macrostates which are labels of k at the "!"-indicated breakpoints, shows that some $\mathcal{A}$-run exists on α with infinitely many visits to F. Hence $\mathcal{A}$ accepts α.

Conversely, suppose that $\mathcal{A}$ accepts α, say by a run ρ which passes through the state $q \in F$ infinitely often. Consider the Safra tree run of $\mathcal{B}$ on α. The root macrostate of each Safra tree in this run is nonempty (since the root macrostate of the i-th Safra tree contains $\rho(i)$). If the root is marked "!" infinitely often (let us call this the "easy case"), then $\mathcal{B}$ accepts by definition and we are done. Otherwise, after the last occurrence of the mark "!" at the root (if marks existed at all), state $q \in F$ will be reached at some later point (being visited infinitely often in ρ) and thus be put into a son macrostate of the root. From this point onwards, the states of the run ρ appear in the macrostates of this son, or (due to horizontal merge operations) get associated to older brothers of this son. Such a shift to an older brother can happen only a finite number of times, after which the states of ρ will be associated to some fixed son of the root; note that the deletion of this son itself by vertical merge is no more possible because the last breakpoint of the root was already passed. If this son is marked "!" again and again, we are done as in the "easy case" before. Otherwise, proceed with this son (in which q occurs infinitely often) in the same way as with the root above; as a consequence, q will occur infinitely often in the macrostates of some fixed *grand*son of the root. Continuing in this way, the "easy case" (and hence acceptance by $\mathcal{B}$) must apply eventually; otherwise the height of the used Safra trees would increase beyond the bound $|Q| - 1$ (which is impossible). Thus $\mathcal{B}$ accepts α. □

The deterministic automaton resulting from Safra's construction is presented more concisely if one refers to the Rabin acceptance condition. The acceptance condition can be formulated as requiring that some node name is missing only finitely often but occurs marked "!" infinitely often. So we get a deterministic Rabin automaton with accepting pairs (E_k, F_k), where E_k contains the Safra trees without node name k and F_k contains the Safra trees with node name k marked "!". In the next proposition we verify that the number of Safra tree nodes k may be bounded by the number of states of the given Büchi automaton, which yields a tight complexity bound for determinization:

Proposition 5.6 *Safra's construction converts a Büchi automaton with n states into a deterministic Rabin automaton with $2^{O(n\cdot\log(n))}$ states and $O(n)$ accepting pairs.*

Proof. Suppose a Büchi automaton with state set $Q = \{q_1, \dots, q_n\}$ is given. In a first step, we verify inductively on the height of Safra trees over Q that the number of nodes in a Safra tree over Q is bounded by n. (This is trivial for height 0; in the induction step observe that the sons of the root define Safra trees of lower height over disjoint sets Q_i of states. Thus, by induction hypothesis, the cardinality of the whole Safra tree is bounded by $(\Sigma_i|Q_i|)+1$, which is $\leq |Q|(= n)$ because $\bigcup_i Q_i$ is a proper subset of Q.) Consequently, as mentioned in the proof above, the numbers $1, \dots, 2n$ are sufficient as node names of Safra trees over Q, and the constructed Rabin automaton has $2n$ accepting pairs. In a second step, note that a state q_i occurring in a Safra tree s belongs to the macrostates along a unique path prefix of s, starting at the root and ending at some node k. A Safra tree is determined if to each q_i this "last node" k is associated (or a dummy value 0 if q_i does not occur in s) and, furthermore, the "parent function", the "next-older-brother function", and the "!-function" on the set of nodes are known. The latter functions associate to each node its parent node, its next-older brother (or 0 if none exists), and 1 or 0 as indicator of presence or absence of "!", respectively. Altogether a Safra tree s is described by four maps, one from $\{q_1, \dots, q_n\}$ to $\{0, \dots, 2n\}$, the three others from $\{1, \dots, 2n\}$ to $\{0, \dots, 2n\}$. The number of combinations of such maps (and hence the number of possible Safra trees) is thus bounded by $(2n + 1)^{n+3\cdot 2n}$, which is in $2^{O(n\cdot\log(n))}$. □

This complexity bound is optimal in the following sense:

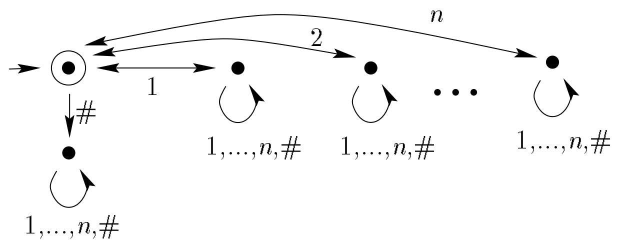
Theorem 5.7 (cf. [Saf88]) *There is no conversion of Büchi automata with n states into deterministic Rabin automata with $2^{O(n)}$ states and $O(n)$ accepting pairs.*

For the proof we use an elegant (and up to now unpublished) example of Michel [Mic88], concerning the complexity of complementing nondeterministic Büchi automata. We present it before giving the proof of Theorem 5.7.

Theorem 5.8 ([Mic88]) *There is a family $(L_n)_{n\geq 1}$ of ω-languages such that each L_n is recognized by a Büchi automaton with $n+2$ states, and any Büchi automaton recognizing the complement of L_n has $\geq n!$ states.*

Proof. We shall define L_n over the alphabet $\{1,\ldots,n,\#\}$ and consider complementation relative to $\{1,\ldots,n,\#\}^\omega$. It is easy (when coding letter i by 0^i1) to adapt the construction to the fixed alphabet $\{0,1,\#\}$, over which the resulting ω-language L'_n is recognized by a Büchi automaton with a number of states linear in n, but such that its complement w.r.t. $\{0,1,\#\}^\omega$ is not Büchi recognizable with $< n!$ states.

Let L_n be the ω-language recognized by the following Büchi automaton:



By the in-out-transitions to and from the final state, which have to be passed infinitely often within any successful run, the following remark is easily shown:

$(*)$ $\alpha \in L_n$ *iff there is a cycle* $(i_1i_2)(i_2i_3)\ldots(i_ki_1)$ *of letter-pairs such that each letter-pair occurs infinitely often as a segment of* α.

Consequently, an ω-word $(i_1\ldots i_n\#)^\omega$ does not belong to L_n for any permutation $(i_1\ldots i_n)$ of $(1\ldots n)$.

Now let $\mathcal{B}$ be a Büchi automaton which accepts the complement language $\{1,\ldots,n,\#\}^\omega \setminus L_n$. Consider any two distinct permutations $(i_1\ldots i_n)$ and $(j_1\ldots j_n)$ of $(1\ldots n)$, so that $\mathcal{B}$ accepts the ω-words $\alpha = (i_1\ldots i_n\#)^\omega$ and $\beta = (j_1\ldots j_n\#)^\omega$. Choose successful runs of $\mathcal{B}$ on α and β, and suppose that in the run on α, the automaton $\mathcal{B}$ finally loops through the set R of states (with some final state, say p), while in the run on β it finally loops through the set S of states. It suffices to show that R and S are disjoint (then $\geq n!$ pairwise disjoint loops exist in $\mathcal{B}$).

For a contradiction assume $q \in R\cap S$. Using the two given runs, we build up a new run through $\mathcal{B}$ which reaches q, loops through R such that an input segment $i_1\ldots i_n$ is traversed at least once and also the final state p is visited, comes back to q, loops through S such that an input segment $j_1\ldots j_n$ is traversed at least once, comes back to q, and so on in alternation through R and S. This run is accepting by its infinitely many visits to p. The corresponding input γ however contains (as we will show) a cycle as described in the characterization $(*)$ of L_n above; thus $\mathcal{B}$ accepts some ω-word in L_n, which gives the desired contradiction.

To verify the existence of a cycle as in $(*)$, consider the first k where the entries i_k, j_k of the two permutations are distinct. Then j_k appears as i_l for some $l > k$, and i_k appears as j_m for some $m > k$. The claimed cycle of letter-pairs occurring infinitely often in γ may now be chosen as $(i_k i_{k+1}), \ldots, (i_{l-1} i_l)$ $(= (i_{l-1} j_k))$, $(j_k j_{k+1}), \ldots, (j_{m-1} j_m)$ $(= (j_{m-1} i_k))$. □

Proof of Theorem 5.7. Assume there is a conversion of Büchi automata into deterministic Rabin automata which transforms Büchi automata with n states into deterministic Rabin automata with $2^{O(n)}$ states and $O(n)$ accepting pairs. Consider the deterministic Rabin automata which would be obtained in this way from the Büchi automata recognizing the languages L_n. We shall convert such a deterministic Rabin automaton $\mathcal{R}$, say with $2^{O(n)}$ states and with accepting pairs (E_i, F_i) $(1 \leq i \leq kn)$, into a nondeterministic Büchi automaton $\mathcal{B}$ which recognizes the complement of L_n and has only $2^{O(n)}$ states, contradicting the previous theorem.

The automaton $\mathcal{B}$ has states of the form q and (q, I, J) where q is a state of $\mathcal{R}$ and I, J are sets of indices from $\{1, \ldots, kn\}$. The first component of such a triple serves to simulate $\mathcal{R}$, the other two to test that $\mathcal{R}$'s acceptance condition fails. This means that a Streett condition holds: For all $i \in \{1, \ldots, kn\}$ there are infinitely many visits to E_i or only finitely many visits to F_i; in other words: Infinitely many visits to F_i imply infinitely many visits to E_i. By nondeterminism, $\mathcal{B}$ guesses at which point the finitely often visited states in a run are all passed ("infinity point"). This is implemented by switching from states q to states (q', I, J), beginning with $I = J = \emptyset$. Afterwards $\mathcal{R}$ collects in the component I the indices i for which visits to F_i occur, similarly in the component J the indices j for which visits to E_j occur. Anytime when $I \subseteq J$ holds, both components are reset to $\emptyset$. This happens infinitely often beyond the infinity point iff for all i, infinitely many visits to F_i imply infinitely many visits to E_i, as was to be checked. Since $\mathcal{B}$ has $2^{O(n)}$ many states, the claim is proved. □

Applications of the Safra determinization construction in obtaining (essentially optimal) complexity bounds for logics of programs are given e.g. in [EJ88]. In [Saf92], Safra achieved a transformation of nondeterministic Streett automata into deterministic Rabin automata with the same asymptotic blow-up as for nondeterministic Büchi automata; more precisely, a nondeterministic Streett automaton with n states and h pairs in the acceptance component is converted into a deterministic Rabin automaton with $2^{O(nh \cdot \log(nh))}$ states and nh pairs. A generalization of the Safra construction to asynchronous finite automata (accepting infinite Mazurkiewicz traces) is presented in [KMS95].

5.3 Applications to Definability and Decision Problems

As a first consequence of McNaughton's Theorem, we note the equivalence between Büchi automata and MSO-logic over infinite words (originally shown by Büchi without use of deterministic automata).

Theorem 5.9 (Büchi's Theorem [Bü62])
An ω-language is Büchi-recognizable iff it is MSO-definable, and the transformation of Büchi automata into MSO-formulas and conversely is effective.

Proof. Given a Büchi automaton $\mathcal{A}$, it is straightforward to formulate acceptance of an input ω-word; the formula of the proof of Theorem 3.1 has to be changed only in the acceptance part (the last conjunctive clause), which should express that infinitely often a final state occurs. For the converse, the proof of Theorem 3.1 is easily copied, using McNaughton's Theorem for the complementation step. $\square$

Corollary 5.10 ([Bü62]) *The theory* S1S *(of all MSO-sentences which are true in the structure $(\omega, S, <)$) is decidable.*

Proof. By Theorem 5.9, a given MSO-sentence φ (without letter predicates Q_a) is effectively transformed into an input-free Büchi automaton $\mathcal{A}$, such that φ is true in $(\omega, S, <)$ iff $\mathcal{A}$ has some successful run. The latter is decidable because a successful run exists iff there is some state q in $\mathcal{A}$ which is reachable from the initial state and such that q is reachable from q via a nonempty path. $\square$

Let us look more closely into the formulas which arise from the proof above and from the application of McNaughton's Theorem. We consider an alphabet $A = \{0,1\}^n$ and a defining MSO-formula $\varphi(X_1, \ldots, X_n)$ (interpreted in ω-words over this alphabet). By Theorem 5.9 it can be rewritten as a formula which describes the acceptance by a Büchi automaton, i.e. in the form

$$\begin{aligned}\exists Y_0 \ldots \exists Y_k \ (I[\overline{Y}(0)] \ \wedge \ & \forall x \forall y (S(x,y) \rightarrow H[\overline{Y}(x), \overline{X}(x), \overline{Y}(y)]) \\ \wedge \ & \forall x \exists y (x < y \wedge K[\overline{Y}(y)]));\end{aligned}$$

here the formula $I[\overline{Y}(0)]$ is a boolean combination of formulas $Y_i(0)$ (using the constant 0 for convenience), and similarly $H[\overline{Y}(x), \overline{X}(x), \overline{Y}(y)]$ and $K[\overline{Y}(y)]$ are boolean combinations of the indicated atomic formulas. We obtain an EMSO formula, or (in the terminology of prefix normal forms) a Σ^1_1-formula.

McNaughton's Theorem yields an additional reduction, from MSO-logic to *weak* MSO-logic, where set quantifiers range only over finite sets. For simplicity, we apply McNaughton's Theorem in the form which yields, given a Büchi automaton as described by the formula above, a deterministic Rabin automaton $\mathcal{R}$, say with a list Ω of accepting pairs $(E_1, F_1), \ldots, (E_m, F_m)$. For each E_i (resp. F_i), consider the usual finite automaton $\mathcal{A}_i$ (resp. $\mathcal{B}_i$) with the same transition graph as $\mathcal{R}$ but final state set E_i (resp. F_i). For each $\mathcal{A}_i$ one can write down a formula $\varphi_i(X_1, \ldots, X_n, y)$ which expresses in an ω-word model $\underline{\alpha}$ that the prefix of α up to (and excluding) y is accepted by $\mathcal{A}_i$. For this, one simply has to relativize each quantifier occurring in the automata normal form for $\mathcal{A}_i$ to the segment $[0, y)$. Formally one replaces a quantifier such as $\exists z \ldots$ by $\exists z(z < y \wedge \ldots)$ and

$\forall z \ldots$ by $\forall z(z < y \to \ldots)$, whence φ_i is called *bounded* in y. Similarly, obtain $\psi_i(X_1, \ldots, X_n, y)$ from $\mathcal{B}_i$, also bounded in y. Hence we have the following result:

Proposition 5.11 *Any MSO-formula $\varphi(X_1, \ldots, X_n)$ is equivalent (over ω-words from $(\{0,1\}^n)^\omega$) to a formula*

$$\bigvee_{i=1}^{m} (\exists x \forall y(x < y \to \neg\varphi_i(X_1, \ldots, X_n, y)) \wedge \forall x \exists y(x < y \wedge \psi_i(X_1, \ldots, X_n, y)))$$

where the φ_i and ψ_i are bounded in y.

Proof. It suffices to note that the $\mathcal{A}_i$ and $\mathcal{B}_i$ introduced above are deterministic and have the same state graphs; thus all formulas φ_i and ψ_i speak indeed about the *same* run on the input ω-word, and the disjunction expresses that $\mathcal{R}$ accepts the input word under consideration. □

When quantifier complexity is measured only in terms of unbounded quantifiers, this result yields a reduction of the Σ_1^1-formulas arising from Büchi automata to boolean combinations of Σ_2^0-formulas. Furthermore, we observe that the set quantifiers in φ_i, ψ_i, which refer to the (finite!) runs of $\mathcal{A}_i$ and $\mathcal{B}_i$, range only over finite sets.

Corollary 5.12 *Any MSO-formula $\varphi(X_1, \ldots, X_n)$ is equivalent (over ω- words) to a weak MSO-formula.*

Proposition 5.11 can be interpreted also in topological terms, referring to the *Cantor topology* on the space of all ω-words over a given alphabet (see the chapter on ω-languages of this Handbook, [Mos80], or [TL94] for definitions). While recognition of an ω-language L by a nondeterministic Büchi automaton shows that L is "projective", the recognition by a deterministic Muller or Rabin automaton puts L into the boolean closure of the second level of the Borel hierarchy.

The disjunctions of Proposition 5.11 lead to a classification of ω-languages, in which the complexity of these formulas (e.g., given by the parameter m) is connected with structural properties of deterministic Muller automata. This theory was initiated by Landweber [Lan69], continued by Staiger and Wagner [SW74], and culminated in a deep structure theory of ω-automata by Wagner [Wag79]. Wagner showed that all deterministic Muller automata accepting a fixed ω-language share a structural invariant, which refers to the chains of strongly connected subsets of the transition graphs (ordered by set inclusion), and is given by the maximal number of alternations between accepting and nonaccepting sets in such a chain. To take a simple example, if the formula of Proposition 5.11 describes a deterministic Büchi automaton (which means that $m = 1$ and only the ψ_1-part of the formula is present), then corresponding Muller automata have systems $\mathcal{F}$ of final (strongly connected) state sets which are upward closed with respect to set inclusion, and thus there are no strongly connected sets $R \subset S$ with $R \in \mathcal{F}$ and $S \notin \mathcal{F}$. As a consequence of this theory, the *Rabin index* of a regular ω-language L is effectively computable, which is the minimal m such that a disjunction of length m as above in Proposition 5.11 defines L. Efficient procedures to determine the Rabin index are developed in [WY95] and [KPB95].

If in Proposition 5.11 we replace the quantifiers $\exists x \forall y(x < y \to \dots)$ by $\forall y \dots$ and $\forall x \exists y(x < y \wedge \dots)$ by $\exists y \dots$, then formulas arise which characterize the Staiger-Wagner-recognizable ω-languages. A beautiful result of [SW74] states that membership of a regular ω-language in this class is decidable and that these ω-languages are precisely those sets L such that L and its complement are both recognized by deterministic Büchi automata.

Another variant of the formulas in Proposition 5.11 is obtained with boolean combinations of statements "there are $\geq k$ segments w" and "there are infinitely many segments w". As in the theory of classical formal languages, the ω-languages defined by such statements are called *locally threshold testable*, and *finitely locally threshold testable* when conditions of the second type are excluded. As for finite words, the finitely locally threshold testable ω-languages coincide with those definable in FO[S]-logic, the first-order logic of successor. Wilke showed in [Wil93] that an ω-language is finitely locally threshold testable iff it is both locally threshold testable and Staiger-Wagner recognizable. Since the latter two properties are decidable (by [BP89] and [SW74]), so is the first, and we may conclude that one can decide effectively whether a regular ω-language is definable in FO[S]-logic.

Büchi's Theorem 5.9 has been refined and extended in many ways. For example, a transfer from ω-words to infinite Mazurkiewicz traces was achieved by Ebinger and Muscholl in [EM96]. In the sequel we discuss in a little more detail two logical systems which are applied in the verification of (nonterminating finite-state) programs, namely *propositional temporal logic* and monadic second-order logic over *timed words*.

Propositional temporal logic PTL is a version of first-order logic over ω-word models where quantifiers over "positions" or "time instances" are captured by temporal operators. One obtains a variable-free notation, reflecting the fact that the reference to such quantified positions is very restricted. The standard operators are X ("next"), F ("eventually"), G ("always"), and U ("until"). PTL-formulas are built up inductively from propositional variables $p_1, p_2, \dots$ by application of boolean connectives, the unary temporal operators X, F, G, and the binary operator U. If the propositional variables $p_1, \dots p_n$ are used, the resulting formulas are interpreted in ω-words over the alphabet $\{0,1\}^n$. To give an idea of the semantics of PTL-formulas, consider the following example:

Example 5.13 The property of ω-words over $\{0,1\}^2$ defined by the condition "after any letter with first component 1 there appears another letter with first component 1 such that between them only letters with second component 0 occur" is formalized by the PTL-formula $\mathrm{G}(p_1 \to \mathrm{X}((\neg p_2)\mathrm{U}p_1))$.

In general, we introduce the semantics of PTL-formulas φ with propositional variables $p_1, \ldots, p_n$ concisely by associating with them certain first-order formulas $\varphi'(X_1, \ldots X_n, x)$, to be interpreted in ω-words over $\{0,1\}^n$ "from position x onwards". (For a more detailed and standard introduction see e.g. [Em90] or [MP92].) For $\varphi = p_i$ we have $\varphi'(x) = X_i(x)$, and the boolean connectives are handled as usual. Given PTL-formulas φ, ψ we set

- $(\mathrm{X}\varphi)'(x) = \exists y(S(x,y) \wedge \varphi'(y))$
- $(\mathrm{F}\varphi)'(x) = \exists y(x \leq y \wedge \varphi'(y))$
- $(\mathrm{G}\varphi)'(x) = \forall y(x \leq y \to \varphi'(y))$
- $(\varphi \mathrm{U} \psi)'(x) = \exists z(x \leq z \wedge \psi'(z) \wedge \forall y(x \leq y < z \to \varphi'(y)))$.

Finally, we say that an ω-word $\alpha \in \{0,1\}^n$ satisfies φ if $(\underline{\alpha}, 0) \models \varphi'(x)$, and an ω-language $L \subseteq \{0,1\}^n$ is called PTL-definable iff for some PTL-formula φ with propositional variables $p_1, \ldots, p_n$ the set L contains precisely those ω-words over $\{0,1\}^n$ which satisfy φ.

By the above definition, each PTL-definable ω-language is first-order definable. A difficult and rather technical result states that the converse is also true:

Theorem 5.14 (Kamp [Kam68], see also [GHR94])
An ω-language is PTL-definable iff it is first-order definable (in the signature with S and $<$).

Despite the practical advantage of short formalizations of interesting properties (see the Example above), a certain weakness of the temporal framework is the fact that the (implicit) quantifications are all unbounded towards infinity, except for the bounded quantification appearing in the until-operator. This makes it hard to formalize properties of finite segments of ω-words, e.g. of finite prefixes. A remedy for this is the introduction of *past operators* which, given a word position as reference point, refer back to the prefix up to this point. In analogy to the "future operators" introduced before one can introduce past operators (namely, "previous", "once", "has always been", and "since"), which allow to express first-order properties of prefixes more easily. If only these temporal operators referring to the past are used, one speaks of a past formula ([MP92]). The use of past formulas makes it possible to put PTL-formulas into a normal form as presented in Proposition 5.11 above. Since in the first-order framework the analogue of this normal form also holds ([Th81]), it turns out that any PTL-formula can be written as a disjunction of formulas $\mathrm{FG}\varphi \wedge \mathrm{GF}\psi$ with past-formulas φ, ψ.

In [MP92], applications of this representation to finite-state program verification are studied.

Another classification of PTL-definable properties is obtained by cancelling certain temporal operators. If the "next"-operator is not admitted, for instance, then only "stutter invariant" ω-languages become definable, in which two ω-words are not distinguished when they can be made equal by shrinking or extending nonzero blocks of identical letters. An interesting class of ω-languages arises by cancelling the "until"-operator from PTL; an automata theoretic (and semigroup theoretic) analysis of this *restricted temporal logic* RTL is carried out in [CPP93]. Recently, an infinite hierarchy based on the nesting of "until"-operators was established in [EW96], providing also further characterizations of restricted temporal logics by structural properties of corresponding automata.

The unidirectional (or "one-way") character of PTL's future operators ("from now to infinity") is useful for the translation of PTL-formulas into ω-automata, essentially because automata also work in a one-way mode. Indeed, for PTL a more direct construction is possible than for general MSO-formulas (or general first-order formulas). It is no more necessary to follow the inductive structure of a given formula, in particular to apply determinization for each negation step (which causes an exponential blow-up each time negation is applied over existential quantification). Instead, for PTL-formulas one can build a Büchi automaton which keeps track of the satisfaction of all subformulas of the given formula simultaneously while reading an input word. (The set of subformulas of a given formula is called its *Fischer-Ladner closure*, and the construction of a model given by truth-values for all subformulas a *Hintikka structure*.) Hence, the state space is essentially the set of truth-value vectors where each component refers to a specific subformula of the given formula. For instance, components referring to complementary subformulas $\neg\psi$ and ψ will have complementary values at each position of a run. Nondeterminism is applied to guess claims about the future correctly, e.g. that a subformula $\mathrm{F}\psi$ or $\varphi\mathrm{U}\psi$ is true; such "obligations" have to be verified at later points in a run. Some book-keeping is necessary for this, which means that auxiliary truth-value components have to be added (however not more than there are subformulas). Altogether, the following result is obtained:

Theorem 5.15 (cf. [LPZ85], [VW94]) *PTL-formulas of length n can be translated effectively into equivalent Büchi automata with $2^{O(n)}$ states (and in time $2^{O(n)}$); consequently, the satisfiability problem for PTL is solvable in exponential time.*

More powerful logics allow the same basic construction, for example the extension of PTL by "automaton operators", which increase the expressive power to capture full MSO-logic (or Büchi automata). The complexity of satisfiability of PTL-formulas is PSPACE-hard ([SC85]); in this sense the bound of the Theorem is optimal.

In program verification, the result is applied for "PTL-model-checking", which means to check that all computation paths of a finite-state program P satisfy a given PTL-formula φ. In automata theoretic terms, one checks that the ω-language of computation paths through P is contained in the ω-language defined by φ. Via the above translation, this can be achieved in a time which is polynomial in the size (number of states) of P and exponential in the length of φ. For more details and for applications in practical verification tasks, the reader should consult specific surveys and monographs such as [Em90], [McM93], [CGL94], [Kur94], [Em96], [Var96].

In practice, the verification of nonterminating systems requires to check more complex computation properties than simply a correct order of events or states in time, as expressible in PTL or MSO-logic. Often, the specification of a program involves also conditions on admissible time intervals or durations of states. There is by now a large number of logics and automata models which incorporate such aspects, e.g. the timed automata of Alur and Dill [AD94]. (For an overview of the field see [AH93].) The underlying models are *timed words*, extending classical ω-words. A timed word is an ω-sequence of letters ("states") together with a sequence of strictly increasing non-negative real numbers, such that the i-th number indicates the beginning of the lifetime of the i-th state. In this framework, a natural extension of Büchi's Theorem is presented by Wilke in [Wil94]; it offers a logic in which time bounds given by natural numbers k are expressible, e.g. statements of the type "there is a time instance $< x$ belonging to a set X such that the time interval from the greatest such instance in X up to x is bounded by k". First-order and monadic second-order quantifications are allowed, with the exception that set variables X used in statements of the type above appear only in a leading block of existential set quantifiers. Wilke showed that this "MSO-logic of relative distance" characterizes the expressive power of the timed automata in the sense of Alur and Dill [AD94]; the decidability of the emptiness problem for these automata implies that also the satisfiability problem for this timed MSO-logic is decidable.

6 Automata and MSO-Logic on Infinite Trees

Rabin showed in [Rab69] that the correspondence between automata and MSO-formulas can be lifted from the domain of infinite words to the domain of infinite trees. As a consequence, the monadic second-order theory S2S of two successor functions turned out to be decidable. The intricate proof as well as its main conclusion, the decidability of a powerful theory, served as starting point of many papers which clarified further the relation between logic and automata and obtained applications in several areas. The core of Rabin's work is a complementation theorem for nondeterministic finite automata on infinite trees. In the first two parts of this section we give a fairly self-contained proof, which follows a game theoretical approach suggested by Büchi [Bü77], [Bü83] and Gurevich and Harrington [GH82], and uses more recent work of [EJ91], [Mst91a], [McN93], [Th95], and [Zie95]. The last section presents some logical applications.

6.1 Automata on Infinite Trees

We shall consider finite tree automata working "top-down" on infinite input trees. Transitions are of the form (q, a, q', q''), allowing to pass from state q at node u with input-tree label a to the states q', q'' at the successor nodes $u0, u1$, respectively. In this way a run is built up. The acceptance condition is a requirement on the state sequences along the paths of the given run, and thus it has the same format as in ω-automata. Again, many different types of acceptance conditions are possible. For the sequel we shall start with the Muller acceptance condition.

Definition 6.1 A *Muller tree automaton* is of the form $\mathcal{A} = (Q, A, q_0, \Delta, \mathcal{F})$ where $Q, A, q_0, \mathcal{F}$ are given as for (sequential) Muller automata, and $\Delta \subseteq Q \times A \times Q \times Q$ is the transition relation. A run of $\mathcal{A}$ on the tree $t \in T_A^\omega$ is a tree $\rho \in T_Q^\omega$, satisfying $\rho(\epsilon) = q_0$ and $(\rho(w), t(w), \rho(w0), \rho(w1)) \in \Delta$ for $w \in \{0,1\}^*$. The run ρ is successful if for each path $\pi \in \{0,1\}^\omega$ we have $\mathrm{In}(\rho|\pi) \in \mathcal{F}$, i.e., along each path of ρ the Muller acceptance condition is satisfied. The automaton $\mathcal{A}$ accepts the tree t if there is a successful run of $\mathcal{A}$ on t. The tree language recognized by $\mathcal{A}$ is the set $T_\omega(\mathcal{A}) = \{t \in T_A^\omega \mid \mathcal{A} \text{ accepts } t\}$.

Other acceptance conditions as known from ω-automata, like the Büchi condition, Rabin condition, Streett condition, are introduced accordingly. It turns out that Muller, Rabin, and Streett tree automata have the same expressive power. (Another approach to acceptance conditions over infinite trees is studied in [BN95]; the requirement that all paths of a run should be successful is replaced there by a condition on the cardinality of the set of successful paths, for instance to be infinite or uncountable.)

Let us look at two simple examples, which also show that Büchi tree automata are strictly weaker than Muller tree automata.

Example 6.2 We describe a Muller tree automaton which recognizes the set

$$T_1 = \{t \in T_{\{a,b\}}^\omega \mid \text{some path through } t \text{ carries infinitely many } b\}.$$

The Muller tree automaton has three states q_0, q_1, q_+ of which q_0, q_1 serve to guess a path down the input tree, such that q_0 signals that a was seen last and q_1 that b was seen last. On nodes outside the guessed path, state q_+ is assumed. Thus, we use the following list of transitions (with $i \in \{0,1\}$): (q_i, a, q_0, q_+), (q_i, a, q_+, q_0), (q_i, b, q_1, q_+), (q_i, b, q_+, q_1), (q_+, a, q_+, q_+), (q_+, b, q_+, q_+). The system of final state sets should then consist of the sets $\{q_0, q_1\}, \{q_1\}, \{q_+\}$. Using the Büchi acceptance condition, it suffices to specify $\{q_1, q_+\}$ as final state set.

Let us see that the complement T_2 of T_1 is recognizable by a Muller tree automaton, however *not* by a Büchi tree automaton ([Rab70]).

Example 6.3 The tree language

$$T_2 = \{t \in T^\omega_{\{a,b\}} \mid \text{each path through } t \text{ carries only finitely many } b\}$$

is Muller recognizable (and hence Rabin recognizable), but not Büchi recognizable: An appropriate (deterministic) Muller tree automaton has two states q_0, q_1 which signal that a, resp. b was seen last (using the transitions (q_i, a, q_0, q_0), (q_i, b, q_1, q_1)). The system of final state sets consists only of $\{q_0\}$. Now for contradiction suppose that T_2 is recognized by a Büchi tree automaton $\mathcal{A}$, say with n states and with final state set F. Consider the input tree t from $T^\omega_{\{a,b\}}$ which has label b exactly at the nodes from 1^+0, $1^+01^+0, \ldots, (1^+0)^n$. Thus label b occurs when a left successor is taken after a sequence of right successors, however allowing at most n left turns. Clearly t belongs to T_2. Consider a successful run ρ of $\mathcal{A}$ on t. Since a final state is visited infinitely often on the path 1^ω of ρ, we may pick m_0 such that $\rho(1^{m_0}) \in F$. Similarly, on the path $1^{m_0}01^\omega$ infinitely many visits to F occur, and thus we may pick m_1 such that $\rho(1^{m_0}01^{m_1}) \in F$. Continuing in this way, we obtain a visit to F at $n+1$ nodes 1^{m_0}, $1^{m_0}01^{m_1}, \ldots, 1^{m_0}01^{m_1}0\ldots 1^{m_n}$. Thus a state repetition must occur, say at nodes u and v from this set. By construction, on the finite path segment of t from u to v the label b occurs (namely, after a left turn). Now form a new input tree t' by repeating this finite path segment from u (inclusive) to v (exclusive) indefinitely, copying also the subtrees which have their roots on this path segment. On the infinite path constructed from these segments, label b occurs infinitely often; thus t' is not in T_2. However, the automaton $\mathcal{A}$ accepts t'; a successful run is easily constructed from ρ using the coincidence of states at nodes u and v. This contradicts the assumption that $\mathcal{A}$ recognizes T_2.

We now turn to the complementation problem for automata on infinite trees. The solution is simplified considerably when we use a seemingly more complicated acceptance condition, the "Rabin chain condition" (introduced by Mostowski [Mst84], [Mst91a]), also called "parity condition" (introduced independently by Emerson and Jutla [EJ91]). The idea is to fix the final state sets not by listing their states separately in each case, but to use a more uniform scheme based on a global indexing of the states. The minimal index of a state within a set of states already determines whether the set as a whole is accepting or not.

Definition 6.4 A *Rabin chain tree automaton* (or *parity tree automaton*) is presented in the form $\mathcal{A} = (Q, A, q_0, \Delta, \Omega)$ where Q, A, q_0, Δ are given as for Muller tree automata, and

$$\Omega: \; E_1 \subset F_1 \subset E_2 \subset F_2 \subset \ldots \subset E_n \subset F_n$$

is a strictly increasing chain of sets of states from Q. A run ρ of the automaton is *successful* if for each path π there is some k such that

$$(*) \qquad \mathrm{In}(\rho|\pi) \cap E_k = \emptyset \text{ and } \mathrm{In}(\rho|\pi) \cap F_k \neq \emptyset.$$

Equivalently, the states in $E_i \setminus F_{i-1}$ are indexed by $2i-1$ and the states in $F_i \setminus E_i$ by $2i$, and a set $\mathrm{In}(\rho|\pi)$ of states is accepting (on the path π) iff the minimal index of states in $\mathrm{In}(\rho|\pi)$ is even (*"parity condition"*).

Rabin chain tree automata are easily converted into Muller tree automata: Given a Rabin chain tree automaton with acceptance component Ω, fix a system $\mathcal{F}$ of final state sets by including all sets F which satisfy the Rabin chain condition $(*)$, applied to F in place of $\mathrm{In}(\rho|\pi)$. The converse is also true ([Mst91b], [Car94]). We give a simple proof, using a data structure of Büchi [Bü83].

Theorem 6.5 *For any Muller tree automaton one can construct an equivalent Rabin chain tree automaton.*

Proof. Let $\mathcal{A} = (Q, A, q_0, \Delta, \mathcal{F})$ be a Muller tree automaton, assuming without loss of generality that $Q = \{1, \ldots, n\}$ and $q_0 = 1$. The states of the desired Rabin chain tree automaton $\mathcal{A}'$ will be permutations of $(1 \ldots n)$ together with an index from $\{1, \ldots, n\}$. This data structure was introduced by Büchi [Bü83] under the name "order-vector with hit". The idea is to keep a record of the states in the order of their "last visits" (as in the "later appearance record" LAR of Gurevich-Harrington [GH82]), together with a pointer to the position where the last change in this record occurred (the "hit position"). In the sequel, we indicate an order-vector with hit h in the form $((i_1 \ldots i_n), h)$, or sometimes more concisely as $(i_1 \ldots \underline{i_h} \ldots i_n)$, where $(i_1 \ldots i_n)$ is a permutation of $(1 \ldots n)$.

Let us explain this data structure by an example. Assume $Q = \{1, 2, 3, 4\}$, and that a sequence 1 3 4 2 3 1 3 3 1... of states over Q is built up, looping finally through the state set $\{1, 3\}$. We start with an order-vector whose last state is 1, indicating that the run over Q begins with 1, and which elsewhere is arbitrary, say $(\underline{2}341)$. The next vector is always obtained by shifting the new momentary state of Q towards the right and setting the hit to the position from where in the previous vector this state was taken. In the example, we obtain, starting from $(\underline{2}341)$, the vectors $(2\underline{4}13)$, $(2\underline{1}34)$, $(\underline{1}342)$, $(2\underline{4}13)$, $(24\underline{3}1)$, $(24\underline{1}3)$, etc. It is clear that in our case where from some point onwards only the states 1, 3 are visited, these two remain at the two last positions of the order-vector, the hit will finally assume only positions 3 and 4, and infinitely often the hit will be on the penultimate position with states 1, 3 in some order listed from there onwards. In general, one verifies the following claim, which allows to extract the set of infinitely often visited states from the information provided by the order-vectors and their hit positions:

Remark 6.6 *Let $j_0j_1j_2\dots$ be a sequence of states from $\{1,\dots,n\}$ and $j'_0j'_1j'_2\dots$ be the corresponding sequence of order vectors with hit positions. Then*

$\mathrm{In}(j_0j_1j_2\dots) = F$ *(say with $|F| = k$) iff the sequence $j'_0j'_1j'_2\dots$ satisfies:*

1. *only finitely often the hit is $< (n-k)+1$,*
2. *infinitely often the hit is $(n-k)+1$, such that the order-vector entries at positions $(n-k)+1,\dots,n$ form the set F.*

This motivates the definition of the desired Rabin chain tree automaton $\mathcal{A}'$ over the set of order-vectors with hit; the indexing of these states by the hit (which amounts to the indexing of final state sets by their cardinality) supplies a scale as needed for introducing the Rabin chain acceptance condition.

The state set of $\mathcal{A}'$ is the set of order-vectors over $Q = \{1,\dots,n\}$ with hit position, the initial state is $(\underline{2}\dots n\ 1)$. If $(i,a,i',i'') \in \Delta$, then all transitions of the following form are put into the transition relation Δ' of $\mathcal{A}'$:

$$(((i_1\dots i_{n-1}i),h),a,((i'_1\dots i'_{n-1}i'),h_1),((i''_1\dots i''_{n-1}i''),h_2)),$$

where $(i'_1\dots i'_{n-1}i')$, $(i''_1\dots i''_{n-1}i'')$ are obtained from $(i_1\dots i_{n-1}i)$ by shifting i', resp. i'', to the right and where h_1 is the position of i' in $(i_1\dots i_{n-1}i)$ and h_2 the position of i'' in $(i_1\dots i_{n-1}i)$. Finally, following the above Remark, the Rabin chain acceptance condition is given by the chain $\Omega := E_1 \subset F_1 \subset \dots \subset E_n \subset F_n$ where E_i is the set of order-vectors with hit $< i$ or of order-vectors with hit i such that the entries from position i onwards do *not* form a set in $\mathcal{F}$; on the other hand, F_i is the union of E_i with the set of all order-vectors with hit i such that the entries from position i onwards *do* form a set in $\mathcal{F}$. It may happen that some difference sets $F_i \setminus E_i$ or $E_{i+1} \setminus F_i$ are empty; if $F_i = E_i$ or if $E_{i+1} = F_i$ then we drop the two sets (F_i and E_i, respectively E_{i+1} and F_i) to ensure that the Rabin chain is proper. It is now easy to check (using the Remark above) that $\mathcal{A}'$ accepts the same trees as $\mathcal{A}$. □

6.2 Determinacy and Complementation

In this section we show that the class of Rabin chain recognizable tree languages is closed under complement.

For this, a game theoretic view of tree automata acceptance is used. With any tree automaton $\mathcal{A} = (Q, A, q_0, \Delta, \Omega)$ and any input tree t one associates an "infinite two-person game" $\Gamma_{\mathcal{A},t}$. It is played by two players, named "Automaton" and "Pathfinder" (following [GH82]), on the tree t. A play of the game is given by an infinite sequence of actions performed by the players in alternation: First Automaton picks a transition from Δ which can serve to start a run at the root of the input tree, then Pathfinder decides on a direction (left or right) to proceed to a son of the root, upon which Automaton chooses again a transition for this node (compatible with the first transition and the input tree); then Pathfinder reacts again by branching left or right from the momentary node, etc. Thus a sequence of transitions (and hence a state sequence from Q) is built up along a path chosen by Pathfinder. Automaton wins the play if the constructed state sequence satisfies the acceptance condition, otherwise Pathfinder wins. Player Automaton tries to realize the acceptance condition, while Pathfinder tries to avoid this.

Formally, it is convenient to describe a play as a sequence of "game positions". A game position where Automaton has to act is a triple of the form (tree node w, tree label $t(w)$, state q at w). By choice of a transition τ of the form $(q, t(w), q', q'')$, a game position of Pathfinder is reached, which is the triple (tree node w, tree label $t(w)$, transition τ at w). Pathfinder's choice of a direction will re-establish a game position for Automaton, consisting of tree node $w0$ or $w1$, the corresponding tree label, and a new state (q', respectively q'', induced by the transition τ chosen before). The standard initial position of the play is Automaton's position $(\epsilon, t(\epsilon), q_0)$.

The game $\Gamma_{\mathcal{A},t}$ is presentable as an infinite graph consisting of all game positions as vertices, such that an edge connects position p_1 to position p_2 if an admissible action transforms p_1 into p_2. Automaton and Pathfinder can be imagined to move in alternation a token through this infinite graph along edges, building up an infinite play.

A *strategy from position* p for the player Automaton, respectively Pathfinder, is a function which for any finite path from p to a position p' (of Automaton, respectively Pathfinder) gives as value a position which is reachable from p' via an edge. A *winning strategy* of Automaton, respectively Pathfinder, from p is a strategy from p which leads to a win of any play, whatever the actions chosen by the adversary (Pathfinder, respectively Automaton) are. A successful run of $\mathcal{A}$ on t immediately yields a winning strategy for Automaton in $\Gamma_{\mathcal{A},t}$: Along each path the suitable choice of transitions is fixed by the run. Conversely, a winning strategy for Automaton in $\Gamma_{\mathcal{A},t}$ clearly provides a method to build up a successful run of $\mathcal{A}$ on t. Thus we reach the following game theoretic formulation of tree automaton acceptance:

Remark 6.7 *The tree automaton $\mathcal{A}$ accepts the input tree t iff in the game $\Gamma_{\mathcal{A},t}$ there is a winning strategy for player Automaton from the initial position $(\epsilon, t(\epsilon), q_0)$.*

Complementation of tree automata means to express the condition that a given automaton $\mathcal{A}$ does not accept t by acceptance of another automaton. In view of Remark 6.7 this means to conclude from nonexistence of a winning strategy for Automaton in $\Gamma_{\mathcal{A},t}$ the existence of a winning strategy for Automaton in a different game $\Gamma_{\mathcal{B},t}$ (such that $\mathcal{B}$ depends only on $\mathcal{A}$ but not on t). For this, we shall proceed in two steps: First we show that if Automaton has no winning strategy in $\Gamma_{\mathcal{A},t}$, then Pathfinder has a winning strategy (from the standard initial position). Secondly, Pathfinder's strategy is converted to an Automaton strategy. The first step means to prove that the games $\Gamma_{\mathcal{A},t}$ are *determined*, i.e., that at least one player has a winning strategy from any given position.

A simple kind of winning strategy will suffice if the tree automaton accepts by the Rabin chain condition, as we assumed. It will turn out that "memoryless" winning strategies are enough. A function is called a *memoryless strategy* if its values depend only on the last positions of the finite initial plays which are given as arguments. In the graph theoretic framework, a memoryless strategy, say for Automaton, is simply given by a subset of the game graph's edge set, such that exactly one outgoing edge remains for any of Automaton's positions.

The above-mentioned first step in the complementation of tree automata is the following result on memoryless determinacy of Rabin chain games, proved in detail later in this section.

Theorem 6.8 (Determinacy of Rabin Chain Tree Automata Games, [EJ91], [Mst91a])
Let $\mathcal{A}$ be a Rabin chain tree automaton and t be an input tree for $\mathcal{A}$. Then in $\Gamma_{\mathcal{A},t}$, from any game position either Automaton or Pathfinder has a memoryless winning strategy.

Let us apply the theorem to establish complementation for Rabin chain tree automata. It will involve the step from a Pathfinder strategy to an Automaton strategy.

Theorem 6.9 (Complementation of Rabin Chain Tree Automata)
For any Rabin chain tree automaton $\mathcal{A}$ over the alphabet A one can construct effectively a Muller tree automaton (and hence also a Rabin chain tree automaton) $\mathcal{B}$ which recognizes $T_A^\omega \setminus T(\mathcal{A})$.

Proof. Let $\mathcal{A} = (Q, A, q_0, \Delta, \Omega)$ be a Rabin chain tree automaton. We have to find a (Muller) tree automaton $\mathcal{B}$ accepting precisely the trees $t \in T_A^\omega$ which are not accepted by $\mathcal{A}$. We start with the following equivalences: For any tree t, $\mathcal{A}$ does not accept t iff (by Remark 6.7) Automaton has no winning strategy from the initial position $(\epsilon, t(\epsilon), q_0)$ in $\Gamma_{\mathcal{A},t}$ iff (by Theorem 6.8)

(+) in $\Gamma_{\mathcal{A},t}$, Pathfinder has a memoryless winning strategy from $(\epsilon, t(\epsilon), q_0)$.

We reformulate (+) in the form "$\mathcal{B}$ accepts t" for some tree automaton $\mathcal{B}$. We start from the observation that Pathfinder's strategy is a function f from the set $\{0,1\}^* \times A \times \Delta$ of his game positions into the set $\{0,1\}$ of directions. Decompose this function into a family $(f_w : A \times \Delta \to \{0,1\})$ of "local instructions", parametrized by $w \in \{0,1\}^*$. The set I of possible local instructions $i : A \times \Delta \to \{0,1\}$ is finite, and thus Pathfinder's winning strategy can be coded by the I-labelled tree s with $s(w) = f_w$. Let $s\hat{\ }t$ be the corresponding $(I \times A)$-labelled tree with $s\hat{\ }t(w) = (s(w), t(w))$ for $w \in \{0,1\}^*$.

Now (+) is equivalent to the following:

> There is an I-labelled tree s such that for all sequences $\tau_0\tau_1\ldots$ of transitions chosen by Automaton and for all (in fact for the unique) $\pi \in \{0,1\}^\omega$ determined by $\tau_0\tau_1\ldots$ via the strategy coded by s, the generated state sequence violates the Rabin chain condition Ω.

A reformulation of this yields:

(1) There is an I-labelled tree s such that $s\hat{\ }t$ satisfies:
 (2) for all $\pi \in \{0,1\}^\omega$
  (3) for all $\tau_0\tau_1\ldots \in \Delta^\omega$
   (4) if the sequence $s|\pi$ of local instructions applied to the sequence of tree labels $t|\pi$ and to the transition sequence $\tau_0\tau_1\ldots$ indeed produces the path π, then the state sequence determined by $\tau_0\tau_1\ldots$ violates Ω.

Condition (4) describes a property of ω-words over $I \times A \times \Delta \times \{0,1\}$ which obviously can be checked by a sequential Muller automaton $\mathcal{M}_4$, independently of t. Condition (3) describes a property of ω-words over $I \times A \times \{0,1\}$, which results from (4) by a universal quantification (equivalently, by a negation, a projection, and another negation). By the established closure properties of Muller recognizable ω-languages, (3) is checked by a sequential and deterministic Muller automaton $\mathcal{M}_3$. Now Condition (2) defines a property of $(I \times A)$-labelled trees, which can be checked by a deterministic Muller *tree* automaton $\mathcal{M}_2$, simulating $\mathcal{M}_3$ along each path. (Note that, by determinism of $\mathcal{M}_3$, the $\mathcal{M}_3$-runs on different paths of an $(I \times A)$-labelled tree agree on the respective common prefix and hence can be merged into one run of $\mathcal{M}_2$.) Finally, applying nondeterminism, a Muller tree automaton $\mathcal{B}$ can be built which checks Condition (1), by guessing a tree s on the input tree t and working on $s\hat{\ }t$ like $\mathcal{M}_2$.

Note that by its construction from $\mathcal{M}_4$, $\mathcal{B}$ does not depend on the tree t under consideration. Thus $\mathcal{B}$ accepts precisely those trees which $\mathcal{A}$ does not accept, as was to be shown. □

It remains to verify the Determinacy Theorem. We refer to the abstract setting of countable game graphs, using terminology and ideas from [GH82], [McN93], [Th95], [Zie95]. The players are now named 0 and 1 (instead of Automaton and Pathfinder).

Definition 6.10 A *game graph* is of the form $G = (V_0, V_1, E, c, C)$, where V_0, V_1 are disjoint at most countable sets of vertices (we always set in this case $V := V_0 \cup V_1$) and $E \subseteq (V_0 \times V_1) \cup (V_1 \times V_0)$ is an edge relation such that for each vertex the set of outgoing edges is nonempty and finite. Furthermore, $c : V \to C$

is a map, called *coloring*, into a finite set C of colors. A *game* is a pair (G, Win) consisting of such a game graph G and an ω-language $Win \subseteq C^\omega$, called *winning set*. The set V_i is intended as the set of game positions where it is the turn of player i to move. A *play* is a sequence $\gamma \in V^\omega$ with $(\gamma(i), \gamma(i+1)) \in E$ for $i \geq 0$. Player 0 wins the play γ if the associated ω-word $c(\gamma(0))c(\gamma(1))\dots$ of colors belongs to Win.

The condition that for each vertex there is an outgoing edge serves to avoid deadlocks in plays. The notions of strategy and winning strategy are defined as before. Recall that a memoryless strategy, say for player 0, is given by a subset of the edge set E which leaves precisely one out-edge for any vertex in V_0.

Example 6.11 Given a Rabin chain tree automaton $\mathcal{A} = (Q, A, q_0, \Delta, \Omega)$, the game $\Gamma_{\mathcal{A},t}$ is of the form above: Take Automaton to be player 0 and Pathfinder to be player 1, and specify the game graph as follows: Let V_0 be the set of triples $(w, t(w), q) \in \{0,1\}^* \times A \times Q$, V_1 the set of triples $(w, t(w), \tau) \in \{0,1\}^* \times A \times \Delta$. Fix the edge relation E in the natural way so that succeeding game positions match and are also compatible with t, and define the color of a triple $(w, t(w), q)$, resp. $(w, t(w), (q, a, q', q''))$, to be the state q. The winning set collects those state sequences which satisfy Ω.

Example 6.12 Given an *input-free* Rabin chain tree automaton $\mathcal{A} = (Q, q_0, \Delta, \Omega)$ with $\Delta \subseteq Q \times Q \times Q$, define a simpler game $\Gamma_{\mathcal{A}}$, in which the tree t and the parameter w in the game positions are suppressed: Let $V_0 = Q$, $V_1 = \Delta$, and fix E in analogy to the previous example, collecting the edges $(q, (q, q', q''))$, $((q, q', q''), q')$, and $((q, q', q''), q'')$ for $(q, q', q'') \in \Delta$. The coloring c is the identity on $V_0 (= Q)$ and maps a transition $(q, q', q'') \in V_1$ to q. The winning set consists again of the state sequences which satisfy Ω. Since the game graph is finite one speaks of a *finite-state game*. As in Remark 6.7 we obtain: Player 0 (Automaton) has a winning strategy in $\Gamma_{\mathcal{A}}$ from position q_0 iff the automaton $\mathcal{A}$ admits at least one successful run.

Theorem 6.13 (Memoryless Determinacy of Rabin Chain Games)
Let $G = (V_0, V_1, E, c, C)$ be a game graph and Win be a winning set specified by a Rabin chain condition, referring to the chain $\Omega : E_1 \subset F_1 \subset \dots \subset E_n \subset F_n \subseteq C$ (i.e., with $\alpha \in Win$ iff $\exists k(\mathrm{In}(\alpha) \cap E_k = \emptyset$ and $\mathrm{In}(\alpha) \cap F_k \neq \emptyset))$. Then from any vertex of G either player 0 or player 1 has a memoryless winning strategy.

An application of this result to the games $\Gamma_{\mathcal{A},t}$ yields the Determinacy Theorem 6.8 and thus the desired complementation of Rabin chain tree automata.

Before turning to the proof, we study the simple case that to win a play over G (with vertex set V) it suffices to reach a certain vertex just *once*. Given a subset $U \subseteq V$ and a player i, the *attractor set* $\mathrm{Attr}_i(G, U)$ is the set of all vertices from where player i can force a visit to some vertex of U in finitely many steps.

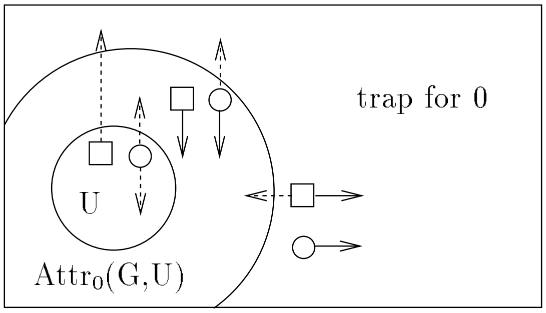
(The suggestive terminology of "attractor sets" and "traps" as used below is due to Zielonka [Zie95].) The following easy lemma shows how to form an attractor set and how to build a memoryless strategy on it which enforces a visit to U; we state it for player 0 (the definition for player 1 is dual). The idea is to collect, inductively for $j = 0, 1, 2, \ldots$, the vertices from which player 0 can force a visit to U in $\leq j$ steps.

Lemma 6.14 (Attractor Lemma)
Let G be a game graph G with vertex set $V = V_0 \cup V_1$ and edge relation E, and suppose $U \subseteq V$. Define a sequence $(U_j)_{j \geq 0}$ by $U_0 = U$ and

$$U_{j+1} = U_j \cup \{u \in V_0 \mid \exists v(E(u,v) \wedge v \in U_j)\} \cup \{u \in V_1 \mid \forall v(E(u,v) \rightarrow v \in U_j)\}$$

Then $\mathrm{Attr}_0(G,U) = \bigcup_{j \geq 0} U_j$. Moreover, a memoryless strategy for player 0 to enforce a visit in U (just once) is obtained by choosing from any V_0-vertex in $U_{j+1} \setminus U_j$ an edge to a vertex in U_j (which exists by construction). If G is finite, $\mathrm{Attr}_0(G,U)$ is the first U_j where $U_j = U_{j+1}$ and hence computable (as is the corresponding strategy to enforce a visit to U).

The figure below illustrates the situation. Vertices in V_0 are indicated by circles, vertices in V_1 by boxes. Arrows denote edges which have to be present, dashed arrows denote edges which may be present.



It is clear that when player i is outside $\mathrm{Attr}_i(G,U)$, he cannot force a transition into $\mathrm{Attr}_i(G,U)$ (otherwise he would already be inside $\mathrm{Attr}_i(G,U)$). Thus the complement Z of a set $\mathrm{Attr}_i(G,U)$ is a *trap* for player i: From $v \in Z \cap V_i$, all edges go back to Z, while from $v \in Z \cap V_{1-i}$ at least one edge goes back to Z. Hence in such a complement set Z each vertex has an outgoing edge back to Z, and we have the following statement:

Remark 6.15 *The complement of an attractor set within the game graph G defines (by the induced subgraph) again a game graph; short: Complements of attractor sets induce subgames.*

Proof of the Determinacy Theorem 6.13. Let $G = (V_0, V_1, E, c, C)$ be a game graph and $Win \subseteq C^\omega$ be defined by the Rabin chain condition with the chain $\Omega : E_1 \subset F_1 \subset \ldots \subset E_n \subset F_n (\subseteq C)$. The claim is proved by induction on the number of nonempty entries of Ω. If no such entry exists, player 1 wins trivially. Assume $E_1 \neq \emptyset$ (otherwise $F_1 \neq \emptyset$; then switch the role of the two players in the remainder of the proof). Note that since E_1 is the smallest set of the chain Ω, infinitely many visits to E_1-colored vertices (short: E_1-vertices) cause a win of player 1: there is no way to cause a win of player 0 by visiting more states!

Let W_0 be the set of vertices from where player 0 has a memoryless winning strategy. The aim is to show that from each vertex in $V \setminus W_0$ player 1 has a memoryless winning strategy.

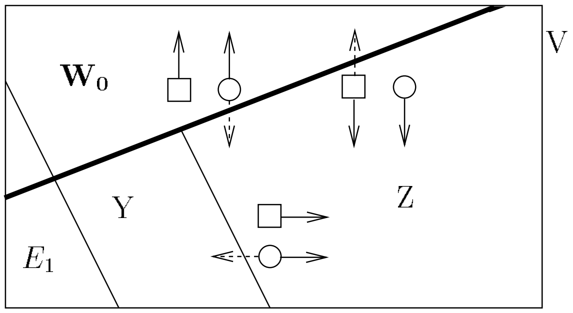
As a preparation, we merge the different memoryless strategies as given from the different vertices in W_0 into a single memoryless strategy which applies uniformly to all vertices in W_0.

Note that a memoryless strategy for player 0 is representable by a graph (U, E_U) where $U \subseteq V$, $E_U \subseteq E \cap (U \times U)$, and E_U has just one outgoing edge from any vertex in $U \cap V_0$. Invoking a well-ordering on the set of those graphs (U, E_U) which constitute winning strategies for player 0, we may index the strategy graphs by ordinal numbers. The desired uniform strategy is now defined on the union of all domains U of these strategy graphs (forming the set W_0), and for any vertex $x \in W_0 \cap V_0$ the chosen out-edge is determined by the unique strategy graph (U, E_U) containing x which has the smallest ordinal index. If we follow this choice of edges during a play, at any moment the index of the used strategy stays equal or decreases. Since a proper decrease of ordinals is possible only a finite number of times, ultimately the relevant index stays constant and hence a fixed of the given winning strategies will be applied, which guarantees that player 0 wins when following the uniform strategy. (For readers who prefer an application of the axiom of choice over the use of well-orderings, the argument starts by choosing one strategy graph (U_v, E_v) for any $v \in W_0$. Since the vertex set is countable, these graphs can be indexed by natural numbers, and the uniform strategy may be defined, for a given $x \in W_0 \cap V_0$, by the unique out-edge as determined by that strategy graph (U_v, E_v) containing x which has minimal index.)

Referring to the uniform strategy on W_0, we see that the complement $V \setminus W_0$ is a trap for player 0 and defines a subgame, denoted $G \setminus W_0$ for short. (Note that by definition of W_0 we have $\mathrm{Attr}_0(G, W_0) = W_0$ and hence Remark 6.15 applies.) Let us assume that some vertices in $V \setminus W_0$ are colored in the minimal set E_1 of the Rabin chain. (Otherwise the induction hypothesis gives the claim of the Theorem easily.) We form the set

$$Y = \mathrm{Attr}_1(G \setminus W_0, (V \setminus W_0) \cap E_1)$$

collecting those vertices in the subgame $G \setminus W_0$ from where player 1 can force a visit to E_1 within this subgame.



Now the complement Z of Y within $V \setminus W_0$ defines again a subgame, being the complement of the attractor set Y. Z is disjoint from the E_1-vertices, whence the induction hypothesis can be applied to Z. Hence we obtain a partition of Z into the vertices from which player 0, resp. player 1 wins over Z by memoryless strategies. If there are indeed vertices from which player 0 wins in Z, player 0 would win from there also relative to the original game over V, contradicting the fact that Z is disjoint from W_0. Thus from each vertex in Z player 1 has a memoryless strategy in the subgame over Z. These strategies can be merged into one uniform strategy over Z, as above for W_0.

This strategy for player 1 over Z is now lifted to yield a memoryless winning strategy for player 1 from all vertices in $V \setminus W_0$: For vertices in the E_1-attractor set Y the memoryless (attractor) strategy to force a visit to an E_1-vertex is applied. When an E_1-vertex within $V \setminus W_0$ is reached, player 1 can be sure to continue by an edge back to $V \setminus W_0$ (recall that $V \setminus W_0$ is a trap for player 0). Thus there can be only two possibilities: Either player 1 is allowed to stay in Z from some moment onwards; then the strategy supplied by the induction hypothesis suffices. Or Z is left infinitely often within $V \setminus W_0$; then player 1 forces visits of E_1-vertices infinitely often by the mentioned memoryless (attractor) strategy, which again causes player 1 to win. □

A determinacy result holds also for games where the winning set is defined by a Muller (or Rabin or Streett) condition. In these cases, the winning strategy of at least one player needs in general some memory (of uniformly bounded finite size), and the construction of strategies is more involved. References on such strategy constructions are [GH82], further developed in [YY90], [Zei94], as well as [Muc92] and [Kla94]. In [Kla94] essentially optimal complexity bounds for complementation of (Streett-) tree automata are given. An approach using *alternating tree automata* was developed by Muller and Schupp [MS90], [MS95]. Alternating automata are a generalization of nondeterministic automata in which transitions are defined by "and-or"-expressions, instead of "or"-expressions as present in nondeterministic automata. In the self-dual framework of alternating automata, complementation is easy, while projection is the nontrivial step. Another natural self-dual calculus to show the complementation of Rabin tree automata is developed by Arnold [Arn94b]; it involves operators for the definition of least and greatest fixed points over the powerset of $\{0,1\}^*$, the set of tree nodes. Definitions of winning strategies in fixed point calculi are presented in [EJ91] and [Wal96]. Fixed point expressions allow very compact representations of the desired vertex sets from where player 0, respectively player 1 wins, but are (as yet) found difficult to read by nonspecialists. Thus we used here a more standard graph theoretic presentation in the style of [McN93], and owing a lot to Zielonka's work [Zie95]. In the exposition above, the problem of introducing memory is settled in advance (following [Th95]): the reduction of Muller tree automata to Rabin chain tree automata of Theorem 6.5, which expands the state space by "order-vectors" (or "later appearance records"), may be viewed as supplying sufficient memory *in* the game graphs. Relative to these expanded game graphs the simple construction of memoryless strategies suffices.

If the game graph is finite, the determinacy result can be sharpened by an effectiveness claim. This is the content of the "Büchi-Landweber Theorem" [BL69], again presented here for the case of the Rabin chain winning condition and memoryless strategies (instead of the classical Muller condition and finite-memory strategies). The proof is simple in the presence of the Determinacy Theorem 6.13.

Theorem 6.16 (Effective Determinacy of Finite-State Games, [BL69])
Let (G, Win) be a game where G is finite and Win is given in Rabin chain form as in the preceding Theorem. Then the sets U_0, U_1 of vertices from which player 0, respectively 1, wins by a memoryless strategy exhaust the vertex set of G and are effectively computable, as well as corresponding memoryless winning strategies (specified by subsets of the edge set of G).

Proof. By Theorem 6.13, each vertex belongs to either U_0 or U_1. We verify that the property of a vertex v of G to belong to U_0 is in NP (and hence of course decidable): Given $G = (V_0, V_1, E, c, C)$ and a vertex v, one guesses a subset of the edge set which defines a strategy for player 0 from vertex v (i.e., has precisely one outgoing edge from any vertex in V_0, keeps all outgoing edges from vertices in V_1, and contains an edge with source v), and then checks that in this "strategy graph" player 1 cannot win. This means that player 1, starting from v, cannot choose edges which allow him to reach (and repeatedly loop through) a cycle that violates the winning condition Win. Clearly this can be tested in polynomial time.

The test whether $v \in U_1$ and the detection of corresponding winning strategies is analogous, with players 0 and 1 exchanged. □

By Theorem 6.13, the complement property of "$v \in U_0$" is "$v \in U_1$". Thus membership in U_0 (as well as membership in U_1) is a problem in NP $\cap$ co$-$NP. It is open whether a polynomial-time algorithm exists. This question is equivalent to the problem whether there is a polynomial-time model-checking algorithm for the modal μ-calculus ([EJS93], [Em96]).

It is possible to avoid the use of the Determinacy Theorem 6.13 and to construct the sets U_0 and U_1 as well as the corresponding winning strategies directly. This is the approach of the (rather difficult) original proof of Büchi and Landweber for finite-state games with Muller winning condition ([BL69], see also [TB73]). The use of the Rabin chain winning condition allows a simpler construction, by an induction on the size of the game graphs (see [McN93, Sect. 6], [Th95]). Theorem 6.16 provides a solution to "Church's Problem" [Chu63], which asked for an automatic synthesis of reactive finite-state programs from automaton specifications (or from MSO-specifications, invoking their translation into automata).

An easy application of Theorem 6.16 shows that the emptiness problem of automata over infinite trees is decidable (here with the Rabin chain acceptance condition). As a preparation, we introduce the notion of a "regular tree" over an alphabet A.

Definition 6.17 A tree $t \in T_A^\omega$ is called *regular* if it is "finitely generated", i.e. generated by a deterministic finite automaton $\mathcal{B} = (Q_\mathcal{B}, \{0,1\}, q_{0\mathcal{B}}, \delta_\mathcal{B}, f_\mathcal{B})$ equipped with an output function $f_\mathcal{B} : Q_\mathcal{B} \to A$. The label $t(w)$ of the tree t at node $w \in \{0,1\}^*$ is $f_\mathcal{B}(\delta_\mathcal{B}(q_{0\mathcal{B}}, w))$, the output of $\mathcal{B}$ after reading input w.

There is an equivalent definition in terms of input-free deterministic tree automata (without acceptance condition). The idea is to capture the inputs $0, 1$ of $\mathcal{B}$ ("directions") by the two branchings which are given within tree automaton transitions. From a finite word automaton $\mathcal{B}$ as above, derive a deterministic tree automaton $\mathcal{C} = (Q_\mathcal{B} \times A, (q_{0\mathcal{B}}, a_0), \Delta)$, setting $a_0 = f_\mathcal{B}(\delta_\mathcal{B}(q_0, \epsilon))$ and allowing a transition $((q_1, a_1), (q_2, a_2), (q_3, a_3))$ in Δ iff $f_\mathcal{B}(q_i) = a_i$ for $i = 1, 2, 3$, $\delta_\mathcal{B}(q_1, 0) = q_2$, and $\delta_\mathcal{B}(q_1, 1) = q_3$. Clearly the unique run of $\mathcal{C}$ generates (in its A-component) the tree which is generated by the word automaton $\mathcal{B}$. Conversely, an input-free tree automaton as above induces canonically a word automaton which generates the same (regular) tree.

Theorem 6.18 (Rabin Basis Theorem, cf. [Rab72])
For Rabin chain tree automata $\mathcal{A}$, the emptiness problem "$T_\omega(\mathcal{A}) = \emptyset$?" is decidable, and any nonempty set $T_\omega(\mathcal{A})$ contains a regular tree (whose generating automaton $\mathcal{B}$ is obtained effectively from $\mathcal{A}$).

Proof. Given a Rabin chain tree automaton $\mathcal{A} = (Q, A, q_0, \Delta, \Omega)$, proceed to the "input-guessing" (and input-free) tree automaton $\mathcal{A}' = (Q \times A, \{q_0\} \times A, \Delta', \Omega')$, which nondeterministically generates an input tree t (by its several initial states and its transitions) and on t works like $\mathcal{A}$ (by an appropriate definition of Δ' and Ω'). Then: $T_\omega(\mathcal{A}) \neq \emptyset$ iff $\mathcal{A}'$ has some successful run.

We consider the finite-state game $\Gamma_{\mathcal{A}'}$ associated to $\mathcal{A}'$ as in Example 6.12. By the game theoretical formulation of acceptance, $\mathcal{A}'$ has some successful run iff in $\Gamma_{\mathcal{A}'}$ the player Automaton wins from some initial position (q_0, a). Whether this holds can be checked effectively by Theorem 6.16, which yields the decidability claim.

Now assume $T_\omega(\mathcal{A}) \neq \emptyset$, i.e., that $\mathcal{A}'$ admits a successful run. So in $\Gamma_{\mathcal{A}'}$ the player Automaton wins from some initial position (q_0, a), and by Theorem 6.16 he does so by means of a memoryless strategy. This strategy induces a deterministic tree automaton as "subautomaton" of $\mathcal{A}'$, where for each state (q, a) (as game position for Automaton) only one transition exists (as move of Automaton) for continuation of a run. By the remark above, such a deterministic tree automaton generates a regular tree. By construction of $\mathcal{A}'$, this regular tree belongs to the tree language recognized by $\mathcal{A}$. $\square$

In Theorem 6.18 we applied the effective determinacy result 6.16. Rabin used a converse approach in [Rab72]; he gave a direct proof of the Basis Theorem (for tree automata with Rabin acceptance condition) and used the existence of regular trees to show that finite-state winning strategies exist in games over finite graphs (see e.g. [Th90]).

In [EJ88] (see also [Em96]) it is proved that the non-emptiness problem for Rabin tree automata with m states and n accepting pairs is solvable in time $O((mn)^{3n})$. Furthermore, a polynomial-time reduction of the propositional satisfiability problem 3-SAT to the non-emptiness problem of Rabin tree automata shows the latter to be NP-complete.

6.3 Applications to Decision Problems of MSO-Logic

The complementation theorem for tree automata is the central step in connecting MSO-formulas and tree automata.

We consider monadic second-order formulas interpreted in the structure $\underline{T} = (\{0,1\}^*, S_0^T, S_1^T)$ of the binary tree, where S_i^T is the i-th successor relation (i.e., $S_i^T(u, v)$ holds iff $ui = v$). The set of sentences (in the corresponding language with the two successor relation symbols S_0, S_1) which are true in $\underline{T}$ form the theory S2S ("second-order theory of two successors"). Monadic second-order formulas $\varphi(X_1, \ldots, X_n)$ with free set variables $X_1, \ldots X_n$ are interpreted in expanded structures $\underline{t} = (\underline{T}, P_1, \ldots, P_n)$. As explained in Section 2.1, such a tree structure $\underline{t}$ is identified with the corresponding infinite tree $t \in T^\omega_{\{0,1\}^n}$; for each node $w \in \{0,1\}^*$ we have $t(w) = (c_1, \ldots, c_n)$ where $c_i = 1$ iff $w \in P_i$.

The equivalence between MSO-logic and tree automata rests on the following statement:

Theorem 6.19 *For any formula $\varphi(X_1, \ldots, X_n)$ of the monadic second-order language in the signature with S_0, S_1, one can construct effectively a Muller tree automaton $\mathcal{A}$ such that $\mathcal{A}$ accepts a tree t iff $\underline{t}$ satisfies φ.*

Proof. Follow the pattern of Theorem 2.1 and consider the modified but equivalent logic MSO_0 in which first-order quantifiers are simulated by second-order quantifiers over singletons. By induction on formulas of this logic one constructs corresponding tree automata. The case of atomic formulas is easy, as are the induction steps concerning $\vee$ and $\exists$ (using nondeterminism). The complementation step is clear from Theorem 6.9. $\square$

By formalizing the Muller (or Rabin or Streett) acceptance condition of tree automata in MSO-logic, tree automata (and hence MSO-formulas) are converted into equivalent Σ_2^1-formulas: A sequence of existential set quantifiers expresses the existence of a run, whereas the condition that the run is successful requires a universal quantifier over paths (i.e., a universal set quantifier) followed by a first-order formula.

More refined results of tree language definability are obtained when restricted MSO-formulas are considered. For example, if only *weak* second-order quantifiers are admitted (ranging over finite sets of tree nodes), a proper subclass of the MSO-definable tree languages (the class of *weakly definable* tree languages) is obtained. As shown by Rabin [Rab70], these tree languages are the sets L such that both L and the complement of L are recognizable by Büchi tree automata. The classification of weak second-order formulas according to quantifier alternation of the prenex normal form yields an infinite hierarchy ([Th82b]). Another hierarchy is built up by classifying the Rabin recognizable tree languages according to the number of disjunction members in the Rabin acceptance condition. Niwiński [Niw88] proved that this hierarchy is infinite, sharpening considerably the separation of Büchi and Rabin recognizability as explained above in Example 6.3. For a more detailed synopsis of the classification of Rabin recognizable tree languages and for further references we refer the reader to the concluding section of [TL94]. The connections to fixed point logics constitute an own fascinating chapter of definability theory and are developed by Arnold and Niwiński in [AN92], [Niw96].

Let us turn to decidability results for monadic second-order theories. An application of Theorem 6.19 to an MSO-sentence φ yields an input-free tree automaton which admits a successful run iff φ is true in the tree structure $\underline{T}$. The existence of such a successful run is decided effectively by Theorem 6.18. Hence we obtain the celebrated

Theorem 6.20 (Rabin Tree Theorem [Rab69]) *The theory* S2S *is decidable.*

Many mathematical theories have been shown to be decidable by an interpretation in S2S; some examples are presented in [Rab69]. In particular, the decidability S2S extends to tree models with arbitrary finite and even countable branching (such trees are easily embedded in the binary tree $\underline{T}$).

Another type of application is the decidability of modal logics or program logics, if their models are propositional Kripke structures, i.e. at most countable directed graphs whose vertices are propositional models. Since any propositional model over say n propositional variables is coded by a vector from $\{0,1\}^n$ (giving a truth value assignment), such a Kripke structure induces (by unravelling) a $\{0,1\}^n$-valued tree t. An embedding of this tree into the binary tree is possible, preserving the prefix relation between tree nodes. (If we reach a t-node v from the root by taking the root's i_1-th successor, from there the i_2-th successor, etc., until reaching v as an i_l-th successor, we code v by the node $1^{i_1}01^{i_2}0\dots 1^{i_l}0$ of the binary tree.) Such an embedding is described by its range, a set $P_0 \subseteq \{0,1\}^*$. Then a Kripke structure over n propositional variables is coded by a binary tree model $(\underline{T}, P_0, P_1, \dots, P_n)$ with $P_1, \dots, P_n \subseteq P_0$. Assume now that any formula φ in n propositional variables of a given modal logic $\mathcal{L}$ can be translated into an S2S-formula $\overline{\varphi}(X_0, X_1, \dots, X_n)$, such that a Kripke structure satisfies φ iff the corresponding tree model $(\underline{T}, P_0, P_1, \dots, P_n)$ satisfies $\overline{\varphi}$. Then satisfiability of $\mathcal{L}$-formulas is reducible to the question whether $\exists X_0 \exists X_1 \dots \exists X_n \overline{\varphi}(X_0, X_1, \dots, X_n)$ holds in $\underline{T}$, which in turn is decidable by Rabin's Tree Theorem. Many modal and temporal logics have been proved decidable along this line; examples are the modal μ-calculus and the computation tree logic CTL* (see e.g. [Th90] or [EJ88] for a more detailed explanation and further references, and [JW95] for a recent automata theoretic study of the modal μ-calculus). Moreover, if a formula of such a logic is satisfiable, i.e. if a binary tree model $(\underline{T}, P_0, P_1, \dots, P_n)$ exists for a corresponding MSO-formula, then, by Rabin's Basis Theorem 6.18, also a *regular* tree model can be guaranteed. Such regular models originate from finite graphs (the generating automata). So the respective modal logic $\mathcal{L}$ has the so-called *finite model property.* Tree automata can also be applied to obtain a solution of the model checking problem for branching-time logics (where satisfaction in a given model is to be tested rather than satisfiability); see [KG96] for a recent study.

The process of unravelling is also the basis of an interesting generalization of Rabin's Tree Theorem. Consider any relational structure $\mathcal{M} = (M, P_1, \dots P_m,\ R_1, \dots, R_n)$ where the P_i are subsets of M and the R_i are binary relations over M. (The restriction to unary and binary relations is not essential but assumed for notational convenience.) The *tree structure* over $\mathcal{M}$ is the structure

$$\mathcal{M}^{\#} = (M^+, S^M, P_1^+, \dots P_m^+, R_1^+, \dots, R_n^+),$$

where M^+ is the set of nonempty sequences over M and for $x, y \in M^+$

- $S^M(x, y)$ iff $x^\wedge m = y$ for some $m \in M$,
- $P_i^+(x)$ iff there are $z \in M^*, m \in M$ with $x = z^\wedge m$ and $P_i(m)$,
- $R_i^+(x, y)$ iff there are $z \in M^*, m, m' \in M$ such that $x = z^\wedge m$, $y = z^\wedge m'$, $R_i(m, m')$.

In unpublished work of Stupp [Stu75], it was shown that the decidability of the monadic second-order theory of a given structure $\mathcal{M}$ can be transferred to

$\mathcal{M}^{\#}$. Rabin's Tree Theorem amounts to the case where $\mathcal{M}$ is the two element structure $(\{0,1\}, P_0, P_1)$ where $P_0 = \{0\}$ and $P_1 = \{1\}$ (which clearly has a decidable monadic second-order theory).

For the unravelling of a structure $\mathcal{M}$, e.g. for the step from a state transition graph to the tree of execution sequences, the above construction does not provide enough information connecting successive tree levels: Here, for a binary relation $R \subseteq M \times M$ we would need a relation $R' \subseteq M^+ \times M^+$ which contains all pairs $(z^\wedge m, z^\wedge m^\wedge m')$ with $R(m, m')$. Given R^+ as above, this relation R' is definable in the presence of an additional unary predicate, the *clone predicate*, defined by

$$C^M = \{x^\wedge m^\wedge m \mid x \in M^+, m \in M\}.$$

Now let the *unravelling* of $\mathcal{M}$ be the structure

$$\mathcal{M}^+ = (M^+, S^M, C^M, P_1^+, \dots P_m^+, R_1^+, \dots, R_n^+).$$

A related notion of unravelling (giving computation trees of deterministic transition systems) is developed in [Cou95]. In unpublished work of Muchnik (see [Sem84]), in [Cou95] and (for the general form) in [Wal96] it is shown how to translate a sentence φ of the monadic second-order language of $\mathcal{M}^+$ into a sentence $\overline{\varphi}$ of the language of the original structure $\mathcal{M}$ such that $\mathcal{M}^+ \models \varphi$ iff $\mathcal{M} \models \overline{\varphi}$. This yields the following powerful transfer theorem for decidability of theories:

Theorem 6.21 (Muchnik, cf. [Wal96]) *If the monadic second-order theory of $\mathcal{M}$ is decidable, so is the monadic second-order theory of $\mathcal{M}^+$.*

A different kind of generalization of the Rabin Tree Theorem is concerned with the monadic second-order theory of infinite graphs which are "regular modifications of trees". A first result in this direction was proved by Muller and Schupp [MS85]; they showed that the monadic second-order theory of any *context-free graph* is decidable. These graphs are obtained as transition graphs of pushdown automata (where a vertex is a word $qv \in Q \cdot P^*$, for a state set Q and a pushdown alphabet P). The binary tree arises as a special case, using the pushdown automaton with a single state q_0 and transitions allowing to add 0 and 1 to the top of any pushdown store content, say with $q_0 0$ as initial configuration.

More general classes of graphs with a decidable monadic second-order theory were obtained by Courcelle [Cou95] and Caucal [Cau96]. We discuss here the graphs considered by Caucal, which are specified by a concrete language theoretical description. Vertices are represented by words over an alphabet A and edges are labelled by letters of an alphabet B; thus a graph is given by its edge set, as a subset of $A^* \times B \times A^*$. The mentioned graphs are formed in three stages, using the notions of a "recognizable graph", "right closure" of a graph, and "rational restriction" of a graph:

Definition 6.22 A graph G, presented as a set of triples $(u, b, v) \in A^* \times B \times A^*$, is called *recognizable* if it is a finite union of sets $U \times \{b\} \times V$ with regular $U, V \subseteq A^*$. Its *right closure*, written $G.A^*$, is obtained from G by including any edge (uw, b, vw) if (u, b, v) belongs to G. A *rational restriction* of a graph H with vertices in A^* *via the regular language* $W \subseteq A^*$ is obtained from H by keeping only the vertices in W and forming the induced subgraph of H. Now let the class $\mathcal{R}$ contain all graphs which are rational restrictions of right closures of recognizable graphs.

Example 6.23 Any transition graph of a pushdown automaton $\mathcal{A}$ belongs to $\mathcal{R}$: Choose the alphabet A to be the union of the state set Q and the pushdown alphabet P of $\mathcal{A}$, and let B be the terminal alphabet of $\mathcal{A}$. The finite transition table of $\mathcal{A}$ determines a finite (and hence recognizable) graph G_0 with edge set contained in $Q \cdot P \times B \times Q \cdot P^*$; now the transition graph G of $\mathcal{A}$ is the right closure of G_0 restricted to all vertices in $Q \cdot P^*$ which are reachable from a designated initial configuration $q_0 v_0$. The rules generating these vertices from $q_0 v_0$ have the form $qaw \to q'uw$ with $q, q' \in Q$, $a \in P$, and $u, w \in P^*$; thus they form a prefix rewriting system (or regular canonical system in the sense of Büchi [Bü64]) and are known to generate a regular language. This shows that G belongs to $\mathcal{R}$.

It can be shown that the graphs in $\mathcal{R}$ are obtained from the full binary tree by two operations, "inverse rational substitution" and an abstract version of "rational restriction" (in a certain analogy to the generation of the context-free languages from the Dyck languages by inverse morphisms and intersection with regular sets). Both operations preserve the decidability of the monadic second-order theory. Thus, by Rabin's Tree Theorem, the following holds:

Theorem 6.24 [Cau96] *Each graph in the class $\mathcal{R}$, i.e. each rational restriction of the right closure of a recognizable graph, has a decidable monadic second-order theory.*

It is possible to include also nonregular features in graphs and still keep the decidability of the monadic theory. For example, as shown by Elgot and Rabin [ER66], there are nonregular sets P of natural numbers, e.g. the set of squares, the set of powers of 2 or the set of factorial numbers, such that the structure (ω, S, P) of the natural numbers with successor and expanded by P has a decidable monadic second-order theory. Nevertheless, slight generalizations of the operations leading to Theorem 6.24 produce graphs with an undecidable monadic second-order theory, for example the infinite grid (with edges $(a^i b^j, a, a^{i+1} b^j)$ and $(a^i b^j, b, a^i b^{j+1})$ for $i, j \geq 0$). So, it seems that Theorems 6.21 and 6.24 exhaust rather well the class of infinite graphs whose monadic second-order theory is decidable.

Acknowledgment

I thank the colleagues and friends of the ESPRIT Working Group ASMICS, who contributed to this work by many helpful questions and remarks.

Special thanks are due to D. Caucal, D. Niwiński, I. Walukiewicz, and W. Zielonka for sending me their as yet unpublished papers and useful hints. Constructive comments by A. Arnold, D. Caucal, B. Courcelle, N. Klarlund, I. Walukiewicz, and W. Zielonka on a pre-final version contributed a lot to improve the text. Finally, I thank the members of the theory group in Kiel for efficient help and support, and G. Rozenberg for his encouragement to write and finish this paper.

References

[AD94] R. Alur, D. Dill, A theory of timed automata, *Theor. Comput. Sci.* **126** (1994), 183-235.

[AH93] R. Alur, T.A. Henzinger, Real-time logics: complexity and expressiveness, *Information and Computation* **104** (1993), 35-77.

[AHU74] A.V. Aho, J.E. Hopcroft, J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass. 1974.

[AN92] A. Arnold, D. Niwiński, Fixed point characterization of weak monadic logic definable sets of trees, in: *Tree Automata and Languages* (M. Nivat, A. Podelski, Eds.), Elsevier Science Publishers, Amsterdam 1992, pp. 159-188.

[Arn94a] A. Arnold, *Finite Transition Systems*, Masson, Paris, and Prentice-Hall, Hemel Hempstead 1994.

[Arn94b] A. Arnold, An initial semantics of the μ-calculus on trees and Rabin's complementation theorem, *Theor. Comput. Sci.* **148** (1994), 121-132.

[BCST88] D.A.M. Barrington, K.J. Compton, H. Straubing, D. Thérien, Regular languages in NC^1, *J. Comput. System Sci.* **38** (1988), 478-499.

[BHMV94] V. Bruyère, G. Hansel, C. Michaux, R. Villemaire, Logic and p-recognizable sets of integers, *Bull. Belg. Math. Soc. Simon Stevin* **1** (1994), 191-238.

[BK95] D. Basin, N. Klarlund, Hardware verification using monadic second-order logic, in: *Computer Aided Verification* (P. Wolper, Ed.), Lecture Notes in Computer Science **939**, Springer-Verlag, Berlin 1995, pp. 31-41.

[BL69] J.R. Büchi, L.H. Landweber, Solving sequential conditions by finite-state strategies, *Trans. Amer. Math. Soc.* **138** (1969), 295-311.

[BN95] D. Beauquier, D. Niwiński, Automata on infinite trees with counting constraints, *Information and Computation* **120** (1995), 117-125.

[BP89] D. Beauquier and J.-E. Pin, Factors of words, in: *Automata, Languages, and Programming, Proc. 16th ICALP* (G. Ausiello et al., Eds.), Lecture Notes in Computer Science **372**, Springer-Verlag, Berlin 1989, pp. 63-79.

[BS95] F. Blanchet-Sadri, Some logical characterizations of the dot-depth hierarchy and applications, *J. Comput. System Sci.* **51** (1995), 324-337.

[Bü60] J.R. Büchi, Weak second-order arithmetic and finite automata, *Z. Math. Logik Grundl. Math.* **6** (1960), 66-92.

[Bü62] J.R. Büchi, On a decision method in restricted second-order arithmetic, in: *Proc. 1960 Int. Congr. for Logic, Methodology and Philosophy of Science*, Stanford Univ. Press, Stanford, 1962, pp. 1-11.

[Bü64] J.R. Büchi, Regular canonical systems, *Arch. Math. Logik Grundlagenforschung* **6** (1964), 91-111.

[Bü77] J.R. Büchi, Using determinacy to eliminate quantifiers, in: *Fundamentals of Computation Theory* (M. Karpinski, Ed.), Lecture Notes in Computer Science **56**, Springer-Verlag, Berlin 1977, pp. 367-378.

[Bü83] J.R. Büchi, State-strategies for games in $F_{\sigma\delta} \cap G_{\delta\sigma}$, *J. Symb. Logic* **48** (1983), 1171-1198.

[Car94] O. Carton, Chain automata, in: *Technology and Applications, Information Processing '94, Vol. I* (B. Pherson, I. Simon, Eds.), IFIP, North-Holland, Amsterdam 1994, pp. 451-458.

[Cau96] D. Caucal, On infinite transition graphs having a decidable monadic theory, in: *Automata, Languages, and Programming, Proc. ICALP'96*, (F. Meyer auf der Heide, B. Monien, Eds.), Lecture Notes in Computer Science, Springer-Verlag, Berlin 1996 (to appear).

[Chu63] A. Church, Logic, arithmetic, and automata, *Proc. Intern. Congr. Math. 1962*, Almqvist and Wiksells, Uppsala 1963, pp. 21-35.

[CG95] C. Choffrut, L. Guerra, Logical definability of some rational trace languages, *Math. Syst. Theory* **28** (1995), 397-420.

[CGL94] E. Clarke, O. Grumberg, D. Long, Verification tools for finite-state concurrent systems, in: *A Decade of Concurrency* (J.W. de Bakker et al., Eds.), Lecture Notes in Computer Science **803**, Springer-Verlag, Berlin 1994, pp. 124-175.

[CPP93] J. Cohen, D. Perrin and J.E. Pin, On the expressive power of temporal logic, *J. Comput. System Sci.* **46** (1993), 271-294.

[Cou90] B. Courcelle, The monadic second-order logic of graphs I: recognizable sets of finite graphs *Inform. and Comput.* **85** (1990), 12-75.

[Cou91] B. Courcelle, The monadic second-order theory of graphs V: on closing the gap between definability and recognizability, *Theor. Comput. Sci.* **80** (1991), 153-202.

[Cou94] B. Courcelle, Monadic second-order definable graph transductions: a survey, *Theor. Comput. Sci.* **126** (1994), 53-75.

[Cou95] B. Courcelle, The monadic second-order theory of graphs IX: Machines and their behaviours, *Theor. Comput. Sci.* **151** (1995), 125-162.

[Cou96] B. Courcelle, The expression of graph properties and graph transformations in monadic second-order logic, in: *Handbook of Graph Transformations, Vol. I: Foundations* (G. Rozenberg, Ed.), World Scientific, Singapore 1996.

[DM96] V. Diekert, Y. Métivier, Partial commutation and traces, in: *Handbook of Formal Language Theory*, Vol. III (G. Rozenberg, A. Salomaa, Eds.), Springer-Verlag, New York (to appear).

[Don70] J. Doner, Tree acceptors and some of their applications, *J. Comput. System Sci.* **4** (1970), 406-451.

[DR95] V. Diekert, G. Rozenberg (Eds.), *The Book of Traces*, World Scientific, Singapore 1995.

[DT90] M. Dauchet, S. Tison, The theory of ground rewrite systems is decidable, *Proc. 5th IEEE Symp. on Logic in Computer Science*, 1990, pp. 242-248.

[EF95] H.D. Ebbinghaus, J. Flum, *Finite Model Theory*, Springer-Verlag, New York 1995.

[EFT94] H.D. Ebbinghaus, J. Flum, W. Thomas, *Mathematical Logic (2nd Ed.)*, Springer-Verlag, New York 1994.

[EH93] J. Engelfriet, H.J. Hoogeboom, X-automata on ω-words, *Theor. Comput. Sci.* **110** (1993), 1-51.

[EJ88] E.A. Emerson, C.S. Jutla, The complexity of tree automata and logics of programs, in: *Proc. 29th IEEE Symp. on Foundations of Computer Science*, 1988, pp. 328-337.

[EJ91] E.A. Emerson, C.S. Jutla, Tree automata, Mu-calculus and determinacy, in: *Proc. 32nd IEEE Symp. on Foundations of Computer Science* (1991), 368-377.

[EJS93] E.A. Emerson, C.S. Jutla, A.P. Sistla, On model checking for fragments of μ-calculus, in: *Computer Aided Verification* (C. Courcoubetis, Ed.), Lecture Notes in Computer Science **697**, Springer-Verlag, Berlin 1993, pp. 385-396.

[Elg61] C.C. Elgot, Decision problems of finite automata design and related arithmetics, *Trans. Amer. Math. Soc.* **98**, (1961), 21-52.

[Em90] E.A. Emerson, Temporal and modal logic, in: *Handbook of Theoretical Computer Science, Vol. B* (J. v. Leeuwen, Ed.), Elsevier Science Publishers, Amsterdam 1990, pp. 995-1072.

[Em96] E.A. Emerson, Automated temporal reasoning about reactive systems, in: *Logics for Concurrency: Structure versus Automata* (F. Moller, G. Birtwistle, Eds.), Lecture Notes in Computer Science **1043**, Springer-Verlag, Berlin 1996, pp. 41-101.

[EM96] W. Ebinger, A. Muscholl, Logical definability on infinite traces, *Theor. Comput. Sci.* **154** (1996), 67-84.

[ER66] C.C. Elgot, M.O. Rabin, Decidability and undefinability of second (first) order theory of (generalized) successor, *J. Symbolic Logic* **31** (1966), 169-181.

[ER93] A. Ehrenfeucht, G. Rozenberg, T-structures, T-functions, and texts, *Theor. Comput. Sci.* **116** (1993), 227-290.

[EW96] K. Etessami, Th. Wilke, An Until hierarchy for temporal logic, in: *Proc. 11th IEEE Symp. on Logic in Computer Science*, 1996 (to appear).

[Fag74] R. Fagin, Generalized first-order spectra and polynomial-time recognizable sets, in: *Complexity of Computation* (R.M. Karp, Ed.), *SIAM-AMS Proceedings* **7** (1974), pp. 43-73.

[FS93] C. Frougny, J. Sakarovitch, Synchronized rational relations of finite and infinite words, *Theor. Comput. Sci.* **108** (1993), 45-82.

[FSV95] R. Fagin, L.J. Stockmeyer, MY. Vardi, On monadic NP vs monadic co-NP, *Information and Computation* **120** (1995), 78-92.

[GHR94] D. Gabbay, I. Hodkinson, M. Reynolds, *Temporal Logic*, Vol. 1, Clarendon Press, Oxford 1994.

[GH82] Y. Gurevich, L. Harrington, Trees, automata, and games, in: *Proc. 14th ACM Symp. on the Theory of Computing*, 1982, pp. 60-65.

[GR96] D. Giammarresi, A. Restivo, Two-dimensional languages, in: *Handbook of Formal Language Theory*, Vol. III (G. Rozenberg, A. Salomaa, Eds.), Springer-Verlag, New York (to appear).

[GRST96] D. Giammarresi, A. Restivo, S. Seibert, W. Thomas, Monadic second-order logic over rectangular pictures and recognizability by tiling systems, *Information and Computation* **125** (1996), 32-45.

[GS84] F. Gécseg, M. Steinby, *Tree Automata*, Akadémiai Kiodó, Budapest 1984.

[Hnf65] W. Hanf, Model-theoretic methods in the study of elementary logic, in: *The Theory of Models* (J. Addison, L. Henkin, P. Suppes, Eds.), North-Holland, Amsterdam 1965, pp. 132-145.

[HP94] H.J. Hoogeboom, P. ten Pas, MSO-definable text languages, in: *Mathematical Foundations of Computer Science 1994* (I. Prívara et al., Eds.), Lecture Notes in Computer Science **841**, Springer-Verlag, Berlin 1994, pp. 413-422.

[HR86] H.J. Hoogeboom, G. Rozenberg, Infinitary languages: basic theory and applications to concurrent systems, in: *Current Trends in Concurrency* (J. de Bakker et al., Eds.), Lecture Notes in Computer Science **224** , Springer-Verlag, Berlin 1986, pp. 266-342.

[Imm87] N. Immerman, Languages that capture complexity classes, *SIAM J. Comput.* **16** (1987), 761-778.

[JW95] D. Janin, I. Walukiewicz, Automata for the modal μ-calculus and related results, in: *Math. Found. of Comput. Sci. 1995* (J. Wiedermann, P. Hájek, Eds.), Lecture Notes in Computer Science **969**, Springer-Verlag, Berlin 1995, pp. 552-562.

[Kam68] J.A. Kamp, *Tense logic and the theory of linear order*, Ph. D. Thesis, Univ. of California, Los Angeles, 1968.

[KG96] O. Kupferman, O. Grumberg, Branching-time temporal logic and tree automata, *Information and Computation* **125** (1996), 62-69.

[Kla94] N. Klarlund, Progress measures, immediate determinacy, and a subset construction for tree automata, *Ann. Pure Appl. Logic* **69** (1994), 243-168.

[KMS95] N. Klarlund, M. Mukund, M. Sohoni, Determinizing Büchi asynchronous automata, in: *Foundations of Software Technology and Theoretical Computer Science* (P.S. Thiagarajan , Ed.), Lecture Notes in Computer Science **1026**, Springer-Verlag, Berlin 1995, pp. 456-470.

[KPB95] S.C. Krishnan, A. Puri, R.K. Brayton, Structural complexity of ω-automata, in: *STACS'95* (E.W. Mayr, C. Puech, Eds.), Lecture Notes in Computer Science **900**, Springer-Verlag 1995, pp. 143-156.

[KS81] T. Kamimura, G. Slutzki, Parallel and two-way automata on directed ordered acyclic graphs, *Inform. Contr.* **49** (1981), 10-51.

[Kur94] R.P. Kurshan, *Computer-Aided Verification of Coordinating Processes*, Princeton University Press, Princeton, N.J. 1994.

[Lad77] R. Ladner, Application of model theoretic games to discrete linear orders and finite automata, *Information and Control* **33** (1977), 281-303.

[Lan69] L.H. Landweber, Decision problems for ω-automata, *Math. Systems Theory* **3** (1969), 376-384.

[LPZ85] O. Lichtenstein, A. Pnueli, L. Zuck, The glory of the past, in: *Logics of Programs* (R. Parikh et al., Eds.), Lecture Notes in Computer Science **193**, Springer-Verlag, Berlin 1985, pp. 196-218.

[LST95] C. Lautemann, Th. Schwentick, D. Thérien, Logics for context-free languages, in: *Computer Science Logic* (L. Pacholski, J. Tiuryn, Eds.), Lecture Notes in Computer Science **933**, Springer-Verlag, Berlin 1995, pp. 205-216.

[McM93] K. McMillan, *Symbolic Model Checking*, Kluwer, Dordrecht 1993.

[McN66] R. McNaughton, Testing and generating infinite sequences by a finite automaton, *Inform. Contr.* **9** (1966), 521-530.

[McN93] R. McNaughton, Infinite games played on finite graphs, *Ann. Pure Appl. Logic* **65** (1993), 149-184.

[McNP71] R. McNaughton and S. Papert, *Counter-Free Automata*, MIT Press, Cambridge, Mass. 1971.

[Mic88] M. Michel, Complementation is more difficult with automata on infinite words, manuscript, CNET, Paris, 1988.

[Mil90] R. Milner, Operational and algebraic semantics of concurrent processes, in: *Handbook of Theoretical Computer Science* (J. v. Leeuwen, Ed.), Elsevier Science Publ., Amsterdam 1990, pp. 1201-1242.

[Mos80] Y. N. Moschovakis, *Descriptive Set Theory*, North-Holland, Amsterdam 1980.

[MP92] Z. Manna, A. Pnueli, *The Temporal Logic of Reactive and Concurrent Programs*, Springer-Verlag, Berlin, Heidelberg, New York 1992.

[MS85] D.E. Muller, P.E. Schupp, The theory of ends, pushdown automata, and second-order logic, *Theor. Comput. Sci.* **37** (1985), 51-75.

[MS90] D.E. Muller, P.E. Schupp, Alternating automata on infinite trees, *Theor.Comput. Sci.* **54** (1987), 267-276.

[MS95] D.E. Muller, P.E. Schupp, Simulating alternating tree automata by non-deterministic automata: new results and new proofs of the theorems of Rabin, McNaughton and Safra, *Theor. Comput. Sci.* **141** (1995), 69-107.

[Mst84] A.W. Mostowski, Regular expressions for infinite trees and a standard form of automata, in: A. Skowron (ed.), *Computation Theory*, Lecture Notes in Computer Science **208**, Springer-Verlag, Berlin 1984, pp. 157-168.

[Mst91a] A.W. Mostowski, Games with forbidden positions, Preprint No. 78, Uniwersytet Gdański, Instytyt Matematyki, 1991.

[Mst91b] A.W. Mostowski, Hierarchies of weak automata and weak monadic formulas, *Theor. Comput. Sci.* **83** (1991), 323-335.

[Muc92] A. Muchnik, Games on infinite trees and automata with dead-ends. A new proof for the decidability of the monadic second-order theory of two successors, *Bull. of the EATCS* **48** (1992), 220-267 (Russian version in *Semiotics and Information* **24** (1984)).

[Mul63] D.E. Muller, Infinite sequences and finite machines, in: *Proc. 4th IEEE Symp. on Switching Circuit Theory and Logical Design*, 1963, pp. 3-16.

[MV96] C. Michaux, R. Villemaire, Presburger arithmetic and recognizability of natural numbers by automata: new proofs of Cobham's and Semenov's theorems, *Ann. Pure Appl. Logic* **77** (1996), 251-277.

[Niw88] D. Niwiński, Fixed points vs. infinite generation, in: *Proc. 3rd IEEE Symp. on Logic in Computer Science*, 1988, pp. 402-409.

[Niw96] D. Niwiński, Fixed points characterization of infinite behaviour of finite state systems, *Theor. Comput. Sci.* (to appear).

[Per90] D. Perrin, Finite Automata, in: *Handbook of Theoretical Computer Science, Vol. B* (J. van Leuwen, ed.), Elsevier Science Publishers, Amsterdam 1990, pp. 1-57.

[Pin86] J.-E. Pin, *Varieties of Formal Languages*, Plenum, New-York, 1986.

[PP86] D. Perrin and J.-E. Pin, First-order logic and star-free sets, *J. Comput. System Sci.* **32** (1986), 393-406.

[Pot95] A. Potthoff, First-order logic on finite trees, in: *TAPSOFT '95* (P.D. Mosses et al., Eds.), Lecture Notes in Computer Science, Springer Verlag, Berlin 1995, pp. 125-139.

[PST94] A. Potthoff, S. Seibert, W. Thomas, Nondeterminism versus determinism of finite automata over directed acyclic graphs, *Bull. Belg. Math. Soc. Simon Stevin* **1** (1994), 285-298.

[PT93] A. Potthoff, W. Thomas, Regular tree languages without unary symbols are star-free, in: *Fundamentals of of Computation Theory* (Z. Esik, Ed.), Lecture Notes in Computer Science **710**, Springer-Verlag, Berlin 1993, pp. 396-405.

[Rab69] M.O. Rabin, Decidability of second-order theories and automata on infinite trees, *Trans. Amer. Math. Soc.* **141** (1969), 1-35.

[Rab70] M.O. Rabin, Weakly definable relations and special automata, in: *Mathematical Logic and Foundations of Set Theory* (Y. Bar-Hillel, ed.), North-Holland, Amsterdam 1970, pp. 1-23.

[Rab72] M.O. Rabin, *Automata on infinite objects and Church's Problem*, Amer. Math. Soc., Providence, RI, 1972.

[Saf88] S. Safra, On the complexity of ω-automata, in: *Proc. 29th IEEE Symp. on Foundations of Computer Science*, 1988, pp. 319-327.

[Saf92] S. Safra, Exponential determinization for ω-automata with strong-fairness acceptance condition, in: *Proc. 24th ACM Symp. on the Theory of Computing*, 1992, pp. 275-282.

[SC85] A.P. Sistla, E.M. Clarke, The complexity of propositional linear time logics, *J. Assoc. Comput. Mach.* **32** (1985), 733-749.

[Sch65] M.P. Schützenberger, On finite monoids having only trivial subgroups, *Information and Control* **8** (1965), 190-194.

[See92] D. Seese, Interpretability and tree automata: a simple way to solve algorithmic problems on graphs closely related to trees, in: *Tree Automata and Languages* (M. Nivat, A. Podelski, Eds.), Elsevier Science Publishers, 1992, pp. 83-114.

[See96] D. Seese, Linear time computable problems and first-order descriptions, *Math. Struct. in Comp. Sci.* 1996.

[Sei92] S. Seibert, Quantifier hierarchies over word relations, in: *Computer Science Logic* (E. Börger et al. Eds.), Lecture Notes in Computer Science **626**, Springer-Verlag, Berlin 1992, 329-338.

[Sem84] A.L. Semenov, Decidability of monadic theories, in: *Proc. MFCS 84* (M.P. Chytil, V, Koubek, Eds.), Lecture Notes in Computer Science **176**, Springer-Verlag, Berlin 1984, pp. 162-175.

[Sim75] I. Simon, Piecewise testable events, *Proc. 2nd GI Conf.*, Springer LNCS 33 (1975), 214-222.

[St82] R.S. Streett, Propositional dynamic logic of looping and converse, *Inform. Contr.* **54** (1982), 121-141.

[Sta87] L. Staiger, Research in the theory of ω-languages, *J. Inf. Process. Cybern. EIK* **23** (1987), 415-439.

[Sta] L. Staiger, ω-languages, in: *Handbook of Formal Language Theory*, Vol. I (G. Rozenberg, A. Salomaa, Eds.), Springer-Verlag, New York (to appear).

[Sti96] C. Stirling, Modal and temporal logics for processes, in: *Logics for Concurrency: Structure versus Automata* (F. Moller, G. Birtwistle, Eds.), Lecture Notes in Computer Science **1043**, Springer-Verlag, Berlin 1996, pp. 149-237.

[Str94] H. Straubing, *Finite Automata, Formal Logic, and Circuit Complexity*, Birkhäuser, Boston, 1994.

[Stu75] J. Stupp, The lattice model is recursive in the original model, manuscript, The Hebrew Univ., Jerusalem 1975.

[STT95] H. Straubing, D. Thérien and W. Thomas, Regular Languages Defined with Generalized Quantifiers, in: *Information and Computation* **118** (1995), 289-301.

[SW74] L. Staiger, K. Wagner, Automatentheoretische und automatenfreie Charakterisierungen topologischer Klassen regulärer Folgenmengen, *Elektron. Informationsverarbeitung u. Kybernetik EIK* **10** (1974), 379-392.

[TB73] B.A. Trakhtenbrot, Y.M. Barzdin, *Finite Automata*, North-Holland, Amsterdam 1973.

[Th81] W. Thomas, A combinatorial approach to the theory of ω-automata, *Information and Control* **48** (1981), 261-283.

[Th82a] W. Thomas, Classifying regular events in symbolic logic, *J. Comput. Syst. Sci.* **25** (1982), 360-375.

[Th82b] W. Thomas, A hierarchy of sets of infinite trees, in: *Theoretical Computer Science* (A.B. Cremers, H.P. Kriegel, Eds.), Lecture Notes in Computer Science **145**, Springer-Verlag, Berlin 1982, pp. 335-342.

[Th84a] W. Thomas, An application of the Ehrenfeucht-Fraïssé game in formal language theory, *Bull. Soc. Math. France, Mem.* **16** (1984), 11-21.

[Th84b] W. Thomas, Logical aspects in the study of tree languages, in: *Ninth Coll. on Trees in Algebra and Programming* (B. Courcelle, Ed.), Cambridge Univ. Press 1984, pp. 31-49.

[Th87] W. Thomas, A concatenation game and the dot-depth hierarchy, in: *Computation Theory and Logic* (E. Börger, Ed.), Lecture Notes in Computer Science **270**, Springer-Verlag, Berlin 1987, pp. 415-426.

[Th90] W. Thomas, Automata on infinite objects, in: *Handbook of Theoretical Computer Science, Vol. B* (J. v. Leeuwen, Ed.), Elsevier Science Publishers, Amsterdam 1990, pp. 135-191.

[Th91] W. Thomas, On logics, tilings, and automata, in: *Automata, Languages, and Programming* (J. Leach et al., Eds.), Lecture Notes in Computer Science **510** , Springer-Verlag, Berlin 1991, pp. 441-453.

[Th95] W. Thomas, On the synthesis of strategies in infinite games, in: *STACS'95* (E.W. Mayr, C. Puech, Eds.), Lecture Notes in Computer Science **900**, Springer-Verlag, Berlin 1995, pp. 1-13.

[TL94] W. Thomas, H. Lescow, Logical specifications of infinite computations, in: *A Decade of Concurrency* (J.W. de Bakker et al., Eds.), Lecture Notes in Computer Science **803**, Springer-Verlag, Berlin 1994, pp. 583-621.

[TW68] J.W. Thatcher, J.B. Wright, Generalized finite automata with an application to a decision problem of second order logic, *Math. Syst. Theory* **2** (1968), 57-82.

[Var96] M.Y. Vardi, An automata-theoretic approach to linear temporal logic, in: *Logics for Concurrency: Structure versus Automata* (F. Moller, G. Birtwistle, Eds.), Lecture Notes in Computer Science **1043**, Springer-Verlag, Berlin 1996, pp. 238-266.

[VW94] M.Y. Vardi, P. Wolper, Reasoning about infinite computations, *Information and Computation* **115** (1994), 1-37.

[Wag79] K.W. Wagner, On ω-regular sets, *Inform. Contr.* **43** (1979), 123-177.

[Wal96] I. Walukiewicz, Monadic second order logic on tree-like structures, in: *STACS'96* (C. Puech, R. Reischuk, Eds.), Lecture Notes in Computer Science **1046**, Springer-Verlag, Berlin 1996, pp. 401-414.

[Wil93] Th. Wilke, Locally threshold testable languages of infinite words, in: *STACS '93* (P. Enjalbert, A. Finkel, K.W. Wagner, Eds.), Lecture Notes in Computer Science **665**, Springer-Verlag, Berlin 1993, pp. 607-616.

[Wil94] Th. Wilke, Specifying timed state sequences in powerful decidable logics and timed automata, in: *Formal Techniques in Real Time and Fault Tolerant Systems* (H. Langmaack et al., Eds.), Lecture Notes in Computer Science **863**, Springer-Verlag, Berlin 1994, pp. 694-715.

[WY95] Th. Wilke, H. Yoo, Computing the Wadge degree, the Lifschitz degree, and the Rabin index of a regular language of infinite words in polynomial time, in: *TAPSOFT'95* (P.D. Mosses et al., Eds.), Lecture Notes in Computer Science **915**, Springer-Verlag, Berlin 1995, 288-302.

[YY90] A. Yakhnis, V. Yakhnis, Extension of Gurevich-Harrington's restricted determinacy theorem: A criterion for the winning player and an explicit class of winning strategies, *Ann. Pure Appl. Logic* **48** (1990), 277-279.

[Zei94] S. Zeitman, Unforgettable forgetful determinacy, *J. Logic Computation* **4** (1994), 273-283.

[Zie87] W. Zielonka, Notes on finite asynchronous automata, *RAIRO Inform. Théor. Appl.* **21** (1987), 99-135.

[Zie95] W. Zielonka, Infinite games on finitely coloured graphs with applications to automata on infinite trees, Rep. 1091-95, LaBRI, Univ. de Bordeaux, to appear in *Theor. Comput. Sci.*.