Classifying Discrete Temporal Properties

Thomas Wilke[⋆]

Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität zu Kiel, D-24098 Kiel, Germany
tw@informatik.uni-kiel.de, http://www.informatik.uni-kiel.de/∼tw/

Abstract. This paper surveys recent results on the classification of discrete temporal properties, gives an introduction to the methods that have been developed to obtain them, and explains the connections to the theory of finite automata, the theory of finite semigroups, and to first-order logic.

The salient features of temporal logic[1] are its modalities, which allow it to express temporal relationships. So it is only natural to investigate how and how much each individual modality contributes to the expressive power of temporal logic. One would like to be able to answer questions like: Can a given property be expressed without using the modality "next"? What properties can be expressed using formulas where the nesting depth in the modality "until" is at most 2?

This survey reports on recent progress on answering such questions, presenting results from the papers [3], [2], [14], [11], and [16] and the thesis [17].

The results fall into three categories: (A) characterizations of fragments of future temporal logic, where a fragment is determined by which future modalities (modalities referring to the future and present only) are allowed in building formulas; (B) characterizations of symmetric fragments, where with each modality its symmetric past/future counterpart is allowed; (C) characterization of the levels of the until hierarchy, where the nesting depth in the "until" modality required to express a property in future temporal logic determines its level.

An almost complete account of the results from category (A) will be given in Sections 2 through 4, including full proofs. These results can be obtained with a reasonable effort in an automata-theoretic framework and the methods used to obtain them are fundamental to the whole subject, in particular, to the results from categories (B) and (C). The results from these two categories are presented in Sections 5 and 6 without going into details of the proofs, which would require a thorough background in finite semigroup theory.

In computer science applications, temporal formulas are interpreted in (finite or infinite) sequences (colored discrete linear orderings), which are nothing else than words (strings or ω-words). Therefore, the set of models of a temporal formula—the property defined by it—can be viewed as a formal language, in

[⋆] Part of the research reported here was conducted while the author was postdoc at DIMACS as part of the Special Year on Logic and Algorithms.

[1] I use "temporal logic" as a synonym for "propositional linear-time temporal logic."

fact, a regular language. In other words, characterizing a fragment of temporal logic amounts to characterizing a certain class of regular languages.

There is a long tradition of classifying regular languages, going back to as early as 1965, when Schützenberger in the seminal paper of the field, [12], characterized the star-free languages as being exactly the ones whose minimal DFA's are counter-free. Given that temporal logic and star-free expressions have the same expressive power (which was only realized much later [5,4,10]), Schützenberger's result also marked the first step in classifying discrete temporal properties: it gave an effective characterization of the class of all regular languages expressible in temporal logic. After an introductory section with terminology and notation, this survey starts off in Section 2 with a new, brief proof that every language recognized by a counter-free DFA is expressible in future temporal logic.

This paper only deals with strings, but most of the results have been extended to ω-words. The reader is referred to the respective original papers.

1 Basic Terminology and Notation

We interpret temporal formulas in strings and use standard notation with regard to strings. The positions of a string of length n are indexed by $0, \dots, n-1$. When u is a string of length n and $0 \leq i \leq j \leq n$, then $u(i,j)$ denotes the string $u(i)u(i+1)\dots u(j-1)$. Further, $u(i,*)$ denotes the suffix $u(i,n)$.

A temporal formula over some alphabet Σ is built from the logical constants $\top$ (true) and $\bot$ (false) and the elements of Σ using the boolean connectives $\neg$ (negation), $\wedge$ (conjunction), and $\vee$ (disjunction) and the *temporal modalities* X (next), F (eventually), and U (until). All connectives and modalities are unary except for $\wedge$, $\vee$, and U, which are binary and written in infix notation. The set of all temporal formulas is denoted by TL.

A *fragment* of temporal logic is a subset of TL obtained by allowing only the use of certain temporal modalities in the construction of formulas. When l is a list of temporal modalities, then TL$[l]$ denotes the respective fragment. For instance, TL$[\mathsf{F}]$ stands for the class of all temporal formulas which can be built from alphabet symbols and the logical constants using boolean connectives and F as the only temporal modality.

Given a temporal formula φ and a string u, one defines what it means for φ to hold in u, denoted $u \models \varphi$. This definition is inductive, where, in particular,

— for every symbol a, $u \models a$ if $u(0) = a$,

— $u \models \mathsf{X}\varphi$ if $|u| > 1$ and $u(1,*) \models \varphi$,

— $u \models \mathsf{F}\varphi$ if there exists i with $0 < i < |u|$ such that $u(i,*) \models \varphi$, and

— $u \models \varphi \,\mathsf{U}\, \psi$ if there exists i with $0 < i < |u|$ such that $u(j,*) \models \varphi$ for every $j \in \{1, \dots, i-1\}$ and $u(i,*) \models \psi$.

Note that $\bot \;\mathsf{U}\; \varphi$ has the same meaning as $\mathsf{X}\varphi$ for any temporal formula φ, and $\top \;\mathsf{U}\; \varphi$ has the same meaning as $\mathsf{F}\varphi$, which means X and F can be derived from U. Sometimes, we will also use the temporal modality G (always), which is another derived modality: it stands for $\neg\mathsf{F}\neg$.

The two modalities F and U have so-called *stutter-invariant counterparts* (for an explanation of the terminology, see Section 4), denoted F_{sf} and U_{sf}, respectively. Their meaning is defined just as above except that i is allowed to be 0 and 0 must also be considered for j. In this regard, the modalities X, F, and U will be referred to as *strict modalities.*

Given a temporal formula φ over some alphabet Σ and an alphabet Γ, we write $\mathcal{L}_\Gamma(\varphi)$ for the set $\{u \in \Gamma^+ \mid u \models \varphi\}$ and say $\mathcal{L}_\Gamma(\varphi)$ is the language over Γ *defined* by φ. (Observe that if Γ is an arbitrary alphabet, φ an arbitrary formula, and ψ the formula obtained from φ by replacing every alphabet symbol not from Γ by $\bot$, then $\mathcal{L}_\Gamma(\varphi) = \mathcal{L}_\Gamma(\psi)$. This means one can always assume that a defining formula only uses symbols from the alphabet of the language in question.) A language is said to be *expressible* in temporal logic (or TL-expressible) if there is a temporal formula that defines it. Similarly, when $\boldsymbol{F}$ is a fragment of temporal logic, a language is expressible in $\boldsymbol{F}$ if there exists a formula in $\boldsymbol{F}$ that defines it.

A *deterministic finite automaton (DFA)* is a tuple $\boldsymbol{A} = (\Sigma, Q, q_I, \delta, F)$ where Σ is a finite alphabet, Q a finite set of *states,* $q_I \in Q$ the *initial state,* $\delta \colon Q \times A \to Q$ the *transition function,* and $F \subseteq Q$ the set of *final states.* The *extended transition function* of $\boldsymbol{A}$, denoted δ^*, is defined by $\delta^*(q, \epsilon) = q$ for $q \in Q$ and $\delta^*(q, ua) = \delta(\delta^*(q, u), a)$ for $q \in Q$, $u \in \Sigma^*$, and $a \in \Sigma$. The language *recognized by* $\boldsymbol{A}$, denoted $\mathcal{L}(\boldsymbol{A})$, is defined by $\mathcal{L}(\boldsymbol{A}) = \{u \in \Sigma^+ \mid \delta^*(q_I, u) \in F\}$. Given a regular language L, the minimal DFA for L is denoted by $\mathbf{A}_L$.

When u denotes a string, then u^ρ denotes the *reverse* of u, i. e., if u is of length n, then $u^\rho = u(n-1)u(n-2)\dots u(0)$. Accordingly, when L denotes a language, then L^ρ denotes the reverse of L, i. e., the language $\{u^\rho \mid u \in L\}$.

2 Full Temporal Logic

It is easy to see that every language expressible in temporal logic is a regular language, i. e., recognizable by a DFA. This raises the question what regular languages are exactly the ones that are expressible in temporal logic. Recall that the minimal DFA recognizing a given regular language is a canonical object to consider when one is interested in classifying a regular language. So more concretely, one can ask for a structural property of DFA's that is enjoyed by the minimal DFA of a given regular language if and only if the language is expressible in temporal logic.

The adequate property is known as counter-freeness. Given a DFA $\boldsymbol{A}$, a sequence $q_0, \dots, q_{m-1}$ of distinct states is a *counter* for a string u if $m > 1$ and $\delta^*(q_i, u) = q_{i+1}$ for $i < m$ where, by convention, $q_m = q_0$. A DFA is *counter-free* if it does not have a counter.

Theorem 1. **[10,4]** *A regular language L is expressible in* TL *if and only if $\mathbf{A}_L$ is counter-free.*

This theorem is a simple consequence of two fundamental results: in 1971, McNaughton and Papert [10] proved that counter-free DFA's recognize exactly the languages that are expressible in first-order logic; in 1980, Gabbay, Pnueli, Shelah, and Stavi [4] showed that temporal logic is as expressive as first-order logic.[2] The latter result is an improvement of a result of Kamp [5] from 1968 that says that temporal logic with future as well as past operators is as expressive as first-order logic in Dedekind-complete orderings.

The difficult implication in Theorem 1 is the one that asserts that a regular language L is expressible in temporal logic if $\mathbf{A}_L$ is counter-free. For this part of the theorem only a few direct proofs have been presented thus far. There is a journal paper by Cohen, Perrin, and Pin [1], Maler's thesis [8], and an accompanying conference paper by Maler and Pnueli [9]. Cohen et al. as well as Maler and Pnueli use some kind of decomposition theory (for finite semigroups or for finite automata); the proof presented below, from [17], avoids such theories.

We need more terminology and notation. A *pre-automaton* is a triple (Σ, Q, δ) where Σ is a finite alphabet, Q a finite set of states, and $\delta\colon Q \times \Sigma \to Q$ a transition function. In other words, a pre-automaton is a DFA without initial and final states. The terminology and notation we have introduced for DFA's transfers to pre-automata in a straightforward way (if applicable). For instance, the extended transition function of a pre-automaton and the property of being counter-free are defined in exactly the same way as for DFA's.

Given a set Q, we view the set Q^Q of all functions on Q as a finite semigroup with composition as product operation. Given $\alpha, \beta\colon Q \to Q$, we write $\alpha\beta$ for the composition of α and β, i. e., for the function given by $q \mapsto \beta(\alpha(q))$. For $\alpha\colon Q \to Q$ and $Q' \subseteq Q$, we write $\alpha[Q']$ for the image of Q' under α, i. e., for $\{\alpha(q) \mid q \in Q'\}$.

Let $\boldsymbol{A} = (\Sigma, Q, \delta)$ be a pre-automaton. For every string $u \in \Sigma^*$ we define its *transformation*, denoted $u^{\boldsymbol{A}}$, as follows. For every $q \in Q$ we set $u^{\boldsymbol{A}}(q) = \delta^*(q, u)$, and we let $S_A = \{u^{\boldsymbol{A}} \mid u \in \Sigma^+\}$. Clearly, this set is closed under functional composition, that is, it is a subsemigroup of Q^Q. It is called the *transformation semigroup* of $\boldsymbol{A}$. For every $\alpha\colon Q \to Q$, we set $L^A_\alpha = \{u \in \Sigma^+ \mid u^{\boldsymbol{A}} = \alpha\}$. Further, $\tilde{L}^A_\alpha$ denotes $L^A_\alpha \cup \{\epsilon\}$ if $\alpha = id_Q$ and else L^A_α.— Observe that if a pre-automaton as above is counter-free and u is a string such that $u^{\boldsymbol{A}}[Q] = Q$, then $u^{\boldsymbol{A}} = id_Q$.

Proof of Theorem 1, from a counter-free DFA to a temporal formula, [17]. We prove that for every pre-automaton $\boldsymbol{A} = (\Sigma, Q, \delta)$ and every $\alpha \in S_A$ the language L^A_α is expressible in temporal logic, which is obviously enough. The proof goes by induction on $|Q|$ in the first place and then on $|\Sigma|$: in the induction step, we will consider pre-automata with the same state space but over a smaller alphabet as well as pre-automata with a smaller state space but over a much larger alphabet.

We distinguish two cases. First, assume there is no symbol $a \in \Sigma$ such that $a^{\boldsymbol{A}}[Q] \subsetneq Q$. Then $a^{\boldsymbol{A}} = id_Q$ for every $a \in \Sigma$, which means $S_A = \{id_Q\}$. This implies $L^A_\alpha = \Sigma^+$ for every $\alpha \in S_A$, and Σ^+ is obviously expressible in temporal logic.

[2] In [4], the authors interpreted temporal logic and first-order logic in ω-words. It is, however, obvious that their result is also valid for strings.

Second, assume $b \in \Sigma$ is such that $b^{\boldsymbol{A}}[Q] \subsetneq Q$. Let $Q' = b^{\boldsymbol{A}}[Q]$, $\Gamma = \Sigma \setminus \{b\}$, and let $\boldsymbol{B}$ be the pre-automaton which results from $\boldsymbol{A}$ by restricting it to the symbols from Γ. Further, let $U_0 = \Gamma^* b$, $\Delta = \{u^{\boldsymbol{A}} \mid u \in U_0\}$, and set $\boldsymbol{C} = (\Delta, Q', \delta')$ where $\delta'(q, \alpha) = \alpha(q)$ for every $q \in Q'$ and $\alpha \in \Delta$. Finally, let $h\colon U_0^+ \to \Delta^+$ be the function defined by $h(u_0 \dots u_{n-1}) = u_0^{\boldsymbol{A}} \dots u_{n-1}^{\boldsymbol{A}}$ for $u_0, \dots, u_{n-1} \in U_0$.

Let $\alpha \in S_A$. We want to show that L_α^A is TL-expressible. To this end, we first partition L_α^A according to how many b's occur in a string; we set

$$L_0 = L_\alpha^A \cap \Gamma^+ \ , \qquad L_1 = L_\alpha^A \cap \Gamma^* b \Gamma^* \ , \qquad L_2 = L_\alpha^A \cap \Gamma^* b \Sigma^* b \Gamma^* \ .$$

Then $L_\alpha^A = L_0 \cup L_1 \cup L_2$. Next, we observe that

$$L_0 = L_\alpha^B \ , \quad L_1 = \bigcup_{\alpha = \beta b^{\boldsymbol{A}} \beta'} \overbrace{\tilde{L}_\beta^B b \tilde{L}_{\beta'}^B}^{L_{\beta,\beta'}} \ , \quad L_2 = \bigcup_{\alpha = \beta b^{\boldsymbol{A}} \gamma \beta'} \overbrace{\tilde{L}_\beta^B b h^{-1}(L_\gamma^C) \tilde{L}_{\beta'}^B}^{L_{\beta,\gamma,\beta'}} \ ,$$

where $\beta, \beta' \in S_B \cup \{id_Q\}$, and $\gamma \in S_C$. Further, we see that

$$L_{\beta,\beta'} = \tilde{L}_\beta^B b \Sigma^* \cap \Gamma^* b \tilde{L}_{\beta'}^B, \quad L_{\beta,\gamma,\beta'} = \tilde{L}_\beta^B b \Sigma^* \cap \Gamma^* b h^{-1}(L_\gamma^C) \Gamma^* \cap \Sigma^* b \tilde{L}_{\beta'}^B, \tag{1}$$

for $\beta, \beta' \in S_B \cup \{id_Q\}$, and $\gamma \in S_C$.

By induction hypothesis, we know that all L_β^B with $\beta \in S_B$ and all L_γ^C with $\gamma \in S_C$ are TL-expressible. It is now a manageable "programming task" to show that under these assumptions all the sets that are intersected on the right-hand sides of the equations in (1) are TL-expressible, which means L_α^A is TL-expressible, as temporal logic is closed under disjunction (union) and conjunction (union). Lemmas 1 and 2 below provide the details. □

Lemma 1. *Let Σ be an alphabet, $b \in \Sigma$, and $\Gamma = \Sigma \setminus \{b\}$. Assume $L \subseteq \Sigma^+$ and $L' \subseteq \Gamma^+$ are TL-expressible. Then so are $\Gamma^* b L$, $\Gamma^* b (L + \epsilon)$, $\Sigma^* b L'$, $\Sigma^* b (L' + \epsilon)$, $L' b \Sigma^*$, and $(L' + \epsilon) b \Sigma^*$.*

Proof. First, let φ and ψ be formulas over Σ and Γ, respectively, such that $\mathcal{L}_\Sigma(\varphi) = L$ and $\mathcal{L}_\Gamma(\psi) = L'$. Then

$$\Gamma^* b L = \mathcal{L}_\Sigma(\neg b \, \mathsf{U}_{\mathsf{sf}} \, (b \wedge \mathsf{X}\varphi)) \ , \qquad \Sigma^* b L' = \mathcal{L}_\Sigma(\mathsf{F}_{\mathsf{sf}}(b \wedge \mathsf{G}\neg b \wedge \mathsf{X}\psi)) \ .$$

The defining formulas for $\Gamma^* b (L + \epsilon)$ and $\Sigma^* b (L' + \epsilon)$ can be obtained in a similar fashion.

Second, we show by induction that for every temporal formula φ over Γ there exists a temporal formula φ^+ such that $\mathcal{L}_\Sigma(\varphi^+) = \mathcal{L}_\Gamma(\varphi) b \Sigma^*$. We can simply set

$$\begin{aligned} a^+ &= a \wedge \mathsf{F} b \ , & (\neg\varphi)^+ &= \neg\varphi^+ \wedge \neg b \wedge \mathsf{F} b \ , \\ (\varphi \wedge \psi)^+ &= \varphi^+ \wedge \psi^+ \ , & (\varphi \, \mathsf{U} \, \psi)^+ &= (\varphi^+ \wedge \neg b) \, \mathsf{U} \, (\psi^+ \wedge \neg b) \ , \end{aligned}$$

where a stands for an arbitrary element of Γ.

Clearly, $\mathcal{L}_\Sigma(\varphi^+ \vee b) = (\mathcal{L}_\Gamma(\varphi) + \epsilon) b \Sigma^*$. □

Lemma 2. *Let Σ, Δ be alphabets, $b \in \Sigma$, $\Gamma = \Sigma \setminus \{b\}$, and $U_0 = \Gamma^* b$. Further, let $h_0 \colon U_0 \to \Delta$ be an arbitrary function and $h \colon U_0^+ \to \Delta^+$ be defined by $h(u_0 \dots u_{n-1}) = h_0(u_0) \dots h_0(u_{n-1})$ for $u_0, \dots, u_{n-1} \in U_0$. For every $d \in \Delta$, let $L_d = \{u \in \Gamma^+ \mid h_0(ub) = d\}$. Assume $L \subseteq \Delta^+$ is expressible in temporal logic and also L_d for every $d \in \Delta$. Then $h^{-1}(L)\Gamma^*$ is expressible in temporal logic.*

Proof. We show by induction that for every temporal formula φ over Δ there exists a temporal formula $\varphi^\#$ over Σ such that $h^{-1}(\mathcal{L}_\Delta(\varphi))\Gamma^* = \mathcal{L}_\Sigma(\varphi^\#)$. For $d \in \Delta$, we either have $h^{-1}(\mathcal{L}_\Delta(d))\Gamma^* = L_d b \Sigma^*$ or $h^{-1}(\mathcal{L}_\Delta(d))\Gamma^* = (L_d + \epsilon) b \Sigma^*$. Thus, the induction basis follows from the previous lemma and the assumption that the languages L_d are TL-expressible. For the induction step, we can set

$$(\neg\varphi)^\# = \neg\varphi^\# \wedge \mathsf{F}_{\mathsf{sf}} b \ , \qquad (\varphi \wedge \psi)^\# = \varphi^\# \wedge \psi^\# \ ,$$
$$(\varphi \,\mathsf{U}\, \psi)^\# = \psi^\# \vee (\varphi^\# \wedge (b \to \mathsf{X}\varphi^\#) \,\mathsf{U}\, (b \wedge \mathsf{X}\psi^\#)) \ . \qquad \square$$

The above proofs are constructive, i. e., following these proofs one can actually construct a temporal formula defining the language recognized by a given counter-free automaton. A closer analysis of the constructions sketched in the proofs yields the following quantitative statement. (Recall that for every preautomaton with n states, the cardinality of its transformation semigroup is $2^{\mathcal{O}(n \log n)}$.)

Corollary 1. *For every counter-free DFA with at most n states and at most m symbols in the alphabet, there exists a temporal formula of size $m\,2^{2^{\mathcal{O}(n \log n)}}$ which defines the language recognized by the DFA.*

3 Strict Fragments

The three basic temporal modalities are X, F, and U. So if we determine fragments of TL by disallowing the use of some of these modalities we obtain eight different fragments. Obviously, some of these have the same expressive power. For instance, the modality X as well as the modality F can be expressed using U only. Thus, all fragments that allow U have the expressive power of full temporal logic:

$$\mathrm{TL}[\mathsf{U}] = \mathrm{TL}[\mathsf{X},\mathsf{U}] = \mathrm{TL}[\mathsf{F},\mathsf{U}] = \mathrm{TL}[\mathsf{X},\mathsf{F},\mathsf{U}] = \mathrm{TL} \ . \qquad (2)$$

By abuse of notation we use an expression like $\mathrm{TL}[\mathsf{X},\mathsf{U}]$ to refer to the specific fragment of TL as well as to the class of languages expressible in this fragment.

The identities in (2) are the only ones that hold: $\mathrm{TL}[\mathsf{X}]$ and $\mathrm{TL}[\mathsf{F}]$ are incomparable in terms of expressive power and both are stronger than $\mathrm{TL}[\,]$ and weaker than $\mathrm{TL}[\mathsf{X},\mathsf{F}]$, which in turn is weaker than full temporal logic.

The aim of this section is to provide structural properties that exactly characterize each of these fragments, just as counter-freeness characterizes expressibility in full temporal logic.

3.1 Forbidden Patterns

We need a convenient way to describe structural properties of DFA's and therefore borrow the notion of "forbidden pattern" from Cohen, Perrin, and Pin [1].[3]

For brevity in notation, given a transition function $\delta\colon Q \times \Sigma \to Q$, we define a product $Q \times \Sigma^* \to Q$ by setting $q\,u = \delta^*(q, u)$ for $q \in Q$ and $u \in \Sigma^*$. Given a set N, an *N-labeled digraph* is a tuple (V, E) where V is an arbitrary set and E a subset of $V \times N \times V$. The *transition graph* of a DFA $\boldsymbol{A} = (\Sigma, Q, q_I, \delta, F)$ is the Σ^+-labeled digraph (Q, E) where $E = \{(q, u, q\,u) \mid q \in Q \text{ and } u \in \Sigma^+\}$. So the transition graph of any DFA is an infinite graph. (It has infinitely many edges, but only finitely many vertices.)

A *pattern* is a labeled digraph whose vertices are *state variables*, usually denoted p, q, ..., and whose edges are labeled with *variables for labels* of two different types: variables for nonempty strings, usually denoted u, v, ..., and variables for symbols, usually denoted a, b, ... In addition, a pattern comes with *side conditions* stating which state variables are to be interpreted by distinct states. We will draw patterns just as we draw graphs. Consider, for instance, Figure 1. In this figure, as well as in all subsequent figures depicting patterns, we adopt the convention that all states drawn solid must be distinct.

We say a Σ^+-labeled digraph *matches a pattern* if there is an assignment to the variables obeying the type constraints and the side conditions so that the digraph obtained by replacing each variable by the value assigned to it is a subgraph of the given digraph.

3.2 Classification Theorem

Using the notion of a forbidden pattern, we can now characterize all fragments:

Theorem 2. [10,4,1,3,11] *Let L be a regular language and $\boldsymbol{F}$ one of the fragments* TL[ ], TL[X], TL[F], TL[X, F], *or* TL. *Then L is expressible in $\boldsymbol{F}$ if and only if the transition graph of $\mathbf{A}_{L^\rho}$ does not match the pattern(s) for $\boldsymbol{F}$ depicted in Figures 1–6.*

Observe that in Figures 1 and 6 the connected graphs are viewed as different patterns (any of which must not occur), whereas Figure 2 shows only one pattern, which happens to be not connected.

The characterizations given in Theorem 2 for TL[ ] and TL[X] are easy to obtain; the characterization for TL is correct because of Theorem 1. The characterization for TL[X, F] was first obtained by Cohen et al. [1]. An alternative proof and a characterization for TL[F] were given in [3], using the same technique for both fragments. In the following two subsections, this technique is demonstrated.

[3] To be precise, what is called a "forbidden pattern" here is referred to as a "forbidden configuration" by Cohen et al.

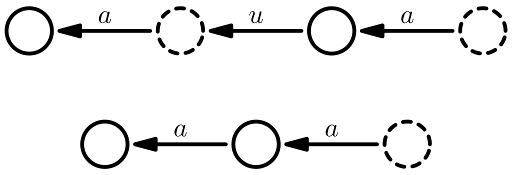

Fig. 1. Patterns forbidden for TL[ ]

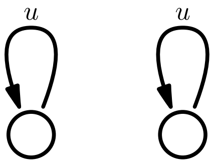

Fig. 2. Pattern forbidden for TL[X]

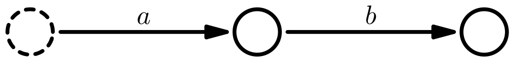

Fig. 3. Pattern forbidden for stutter invariance

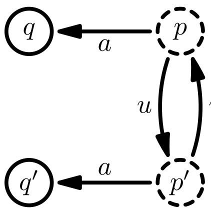

Fig. 4. Pattern forbidden for TL[F]

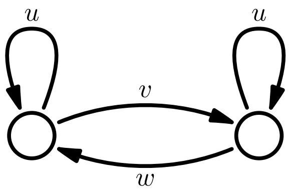

Fig. 5. Pattern forbidden for TL[X, F]

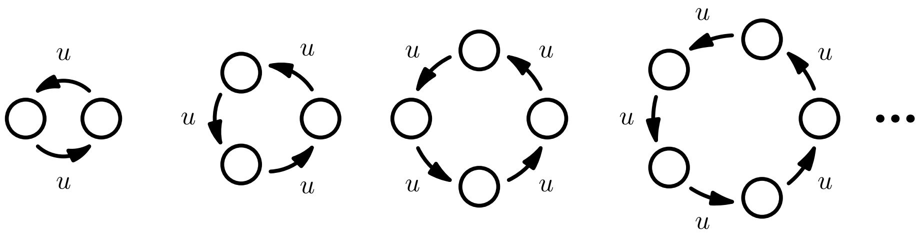

Fig. 6. Patterns forbidden for TL

3.3 Ehrenfeucht-Fraïssé Games

Ehrenfeucht-Fraïssé (EF) games are a standard tool in mathematical logic to tackle questions about the expressive power of a logic. They allow one to reduce such questions to questions about the existence of strategies in specific two-player games, abstract away syntactical peculiarities, and thus represent the combinatorial core of the problems. In our situation, we will use specifically tailored EF games to prove correct the characterizations for TL[F] (and TL[X, F]) given in Theorem 2.

An EF game for TL[F] is played by two players, *Spoiler* (male) and *Duplicator* (female), on a pair of nonempty strings and proceeds in several rounds. The number of rounds to be played is fixed in advance. In each round, a prefix of each of the two strings is chopped off according to a rule explained below so that the outcome of a round is a new pair of strings or an early win for one of the players if the other cannot act according to the rule. Before each round and after the last round, a referee checks if the two strings start with the same symbol. If this is not the case, the referee calls Spoiler the winner of the game. If after the last round Spoiler has not yet won the game, Duplicator is announced the winner. The rule for carrying out a round is as follows. First, Spoiler replaces one of the two strings by a proper, nonempty suffix of it. Then Duplicator replaces the other string by a proper, nonempty suffix of it. If Spoiler cannot follow this rule because both strings have no proper, nonempty suffix (i. e., if both strings are of length 1), he looses, and if Duplicator cannot reply according to the rules because the other string is of length 1, then Spoiler wins.

The idea behind the game is that Spoiler tries to exhibit a difference between the two strings the game starts with whereas Duplicator tries to show they are similar. This can also be phrased in a formal way: Spoiler has a winning strategy in a k-round game if and only if there is a formula φ of "F depth" at most k that holds for one of the two strings but not for the other. The theorem that we will use is the following.

Theorem 3. [3] *Let L be a language. Then L is expressible in* TL[F] *if and only if there exists a number k such that for every pair (u, v) with $u \in L$ and $v \notin L$, Spoiler has a winning strategy in the k-round game on (u, v).*

3.4 Characterization of TL[F]

The claim that a language L is expressible in TL[F] if and only if the transition graph of $\mathbf{A}_{L^\rho}$ does not match the pattern depicted in Figure 4 follows directly from Lemmas 3 and 4 below.

Lemma 3. *Let L be a regular language such that the transition graph of $\mathbf{A}_{L^\rho}$ matches the pattern depicted in Figure 4. Then L is not expressible in* TL[F].

Proof. Let $\mathbf{A}_{L^\rho} = (\Sigma, Q, q_I, \delta, F)$ and assume a, u, and v are chosen so that the pattern in Figure 4 is matched. By minimality of $\mathbf{A}_{L^\rho}$, there exist $x, y \in \Sigma^*$ such that $x(uv)^l uay \in L^\rho$ iff $x(uv)^l ay \notin L^\rho$, for every $l \geq 0$. We show that for $l \geq k \geq 0$ and any choice of strings $x, y \in \Sigma^*$, $u, v \in \Sigma^+$, Duplicator wins the k-round game on $(x(uv)^l uay)^\rho$ and $(x(uv)^l ay)^\rho$. Thus, by Theorem 3, L cannot be expressible in TL[F].

First of all, observe that playing on the first $|ay|$ positions of the two strings does not help Spoiler to win the game: Duplicator will simply copy Spoiler's moves. It is therefore sufficient to show that Duplicator wins the k-round game on $(x(uv)^{l+1}u')^\rho$ and $(x(uv)^l u')^\rho$ for $l \geq k \geq 0$ and any choice of strings $x \in \Sigma^*$, $u, u', v \in \Sigma^+$ where u' is a prefix of u.

The proof of this claim is by induction on k. The induction base, $k = 0$, is trivial. For the inductive step, assume $k > 0$. Write s and t for $(x(uv)^{l+1}u')^\rho$ and $(x(uv)^l u')^\rho$. First, suppose Spoiler removes a prefix of length i from t. Then Duplicator replies by removing a prefix of length $i + |uv|$ from s, and the remaining strings will be identical. Second, assume Spoiler removes a prefix from s, say of length i. If $i > |uv|$, then Duplicator removes the prefix of length $i - |uv|$ from t, and the remaining strings will be identical. If $i \leq |uv|$, then Duplicator removes the prefix of length i from t, and the induction hypothesis applies for the following reason. The remaining strings are $(x(uv)^{l+1}u'')^\rho$ and $(x(uv)^l u'')^\rho$ with $u'' \in \Sigma^+$ a prefix of u, or $(xu(vu)^l v')^\rho$ and $(xu(vu)^{l-1}v')^\rho$ with v' a prefix of v, or $(x(uv)^l u'')^\rho$ and $(x(uv)^{l-1}u'')^\rho$ with $u'' \in \Sigma^+$ a prefix of u. □

For the other direction we need some more notation and terminology. First, we write $\mathrm{SCC}(q)$ for the *strongly connected component* of a node q in a given digraph. Second, given a DFA $\boldsymbol{A} = (\Sigma, Q, q_I, \delta, F)$ and a string $u \in \Sigma^*$, the *rank* of u (with respect to $\boldsymbol{A}$), denoted $\mathrm{rk}(u)$, is the cardinality of the set $\{\mathrm{SCC}(q_I\, u(0,0)), \ldots, \mathrm{SCC}(q_I\, u(0, |u|-2))\}$.

Lemma 4. *Let $\boldsymbol{A}$ be a DFA over some alphabet Σ whose transition graph does not match the pattern depicted in Figure 4. Then $\mathcal{L}(\boldsymbol{A})^\rho$ is expressible in* $\mathrm{TL}[\mathsf{F}]$.

Proof. We prove that if u and v are nonempty strings over Σ such that $q_I\, u \neq q_I\, v$, then Spoiler wins the $(\mathrm{rk}(u)+\mathrm{rk}(v))$-round game on u^ρ and v^ρ, by induction on $\mathrm{rk}(u) + \mathrm{rk}(v)$.

Write $u = u'a$ and $v = v'b$ for appropriate $a, b \in \Sigma$. If $a \neq b$, then Spoiler wins immediately. So in the rest, assume $a = b$. Write p and q for $q_I\, u'$ and $q_I\, v'$. Clearly, $\mathrm{SCC}(p) \neq \mathrm{SCC}(q)$ in the transition graph of $\boldsymbol{A}$, because otherwise it would match the pattern depicted in Figure 4. There are three situations that we distinguish.

1. Neither $\mathrm{SCC}(p)$ is reachable from $\mathrm{SCC}(q)$ nor vice versa.
2. $\mathrm{SCC}(p)$ is reachable from $\mathrm{SCC}(q)$, but $\mathrm{SCC}(q)$ is not reachable from $\mathrm{SCC}(p)$.
3. The same as 2. with the roles of p and q exchanged.

First, assume we are in situation 1. Then it is not possible that q_I belongs to both $\mathrm{SCC}(p)$ and $\mathrm{SCC}(q)$, say it does not belong to $\mathrm{SCC}(p)$. Let i be minimal such that $q_I\, u(0,i) \in \mathrm{SCC}(p)$ and set $p' = q_I\, u(0,i)$. Spoiler replaces u^ρ by $u(0,i)^\rho$. Duplicator either looses immediately (because v is of length 1) or she replies by removing a prefix of v^ρ, say she replaces v^ρ by $v(0,j)^\rho$. Set $q' = q_I\, v(0,j)$. If we had $p' = q'$, then $\mathrm{SCC}(q)$ would be reachable from $\mathrm{SCC}(p)$ — a contradiction. Hence, $p' \neq q'$. By the minimality of i, we also have $\mathrm{SCC}(q_I\, u(0,i-1)) \neq \mathrm{SCC}(p)$, which means $\mathrm{rk}(u(0,i)) < \mathrm{rk}(u)$ and, in particular, $\mathrm{rk}(u(0,i)) + \mathrm{rk}(v(0,j)) < \mathrm{rk}(u)+\mathrm{rk}(v)$, so that the induction hypothesis applies. Spoiler wins the remaining game with one round less.

Second, assume we are in situation 2. Choose i as above. Spoiler does the same as before. Duplicator either looses immediately or she removes a prefix from v^ρ, say she replaces v^ρ by $v(0,j)^\rho$. If we had $q_I\, u(0,i) = q_I\, v(0,j)$, then $\mathrm{SCC}(q)$ would be reachable from $\mathrm{SCC}(p)$ — a contradiction. Just as before, we can apply the induction hypothesis. Situation 3 is symmetric to situation 2. □

Exactly the same technique works for proving the correctness of the characterization of $\mathrm{TL}[\mathsf{X},\mathsf{F}]$. In EF games for this fragment, the additional temporal modality is accounted for by an additional type of round, so-called X rounds. In such a round, Spoiler first chops off the first symbol of one the two strings and Duplicator then chops off the first symbol of the other string. For details, see [3].

4 Stutter-Invariant Fragments

In Section 1 we have defined the so-called stutter-invariant counterparts of F and U, namely $\mathsf{F_{sf}}$ and $\mathsf{U_{sf}}$. In this section, we will obtain effective characterizations for the *stutter-invariant fragments,* $\mathrm{TL}[\mathsf{F_{sf}}]$ and $\mathrm{TL}[\mathsf{U_{sf}}]$. (Observe that $\mathrm{TL}[\mathsf{U_{sf}}] = \mathrm{TL}[\mathsf{F_{sf}},\mathsf{U_{sf}}]$ and $\mathrm{TL}[\mathsf{X},\mathsf{F_{sf}}] = \mathrm{TL}[\mathsf{X},\mathsf{F}]$.)

Strings u and v are *stutter-equivalent* if they both belong to a language of the form $a_0^+ a_1^+ \dots a_k^+$ for some k and appropriate symbols a_i. We use $\equiv_{st}$ to denote stutter equivalence, and it is easy to see that $\equiv_{st}$ is in fact an equivalence relation. A language is *stutter-invariant* if whenever u and v are stutter-equivalent strings, then either u and v belong to this language or u and v do not belong to it, i. e., if this language is a union of stutter equivalence classes.

Lamport [7] observed that every language expressible in $\mathrm{TL}[\mathsf{F_{sf}},\mathsf{U_{sf}}]$ is stutter-invariant. This explains why $\mathsf{F_{sf}}$ and $\mathsf{U_{sf}}$ are called stutter-invariant. Below, we prove that the converse of Lamport's observation holds true as well, in the following sense.

Theorem 4. [3,17] *Let $\boldsymbol{F}$ be one of the stutter-invariant fragments* $\mathrm{TL}[\mathsf{F_{sf}}]$ *and* $\mathrm{TL}[\mathsf{U_{sf}}]$ *and let $\boldsymbol{F}'$ be its strict counterpart,* $\mathrm{TL}[\mathsf{F}]$ *respectively* $\mathrm{TL}[\mathsf{U}]$. *Assume L is an arbitrary language. Then L is expressible in $\boldsymbol{F}$ if and only if L is expressible in $\boldsymbol{F}'$ and stutter-invariant.*

Observe that a regular language L is stutter-invariant if and only if the transition graph of $\mathbf{A}_{L^\rho}$ (or, equivalently, of $\mathbf{A}_L$) does not match the pattern depicted in Figure 3. Thus, the above theorem (together with the classification theorem from the previous section) immediately leads to characterizations of $\mathrm{TL}[\mathsf{F_{sf}}]$ and $\mathrm{TL}[\mathsf{U_{sf}}]$ in terms of forbidden configurations.

Using the characterization results we have obtained so far, one can prove:

Corollary 2. *For every fragment (strict or stutter-invariant) $\boldsymbol{F}$ of temporal logic, the following problem is PSPACE-complete. Given a temporal formula φ, decide whether φ is equivalent to a formula in $\boldsymbol{F}$?*

The upper bound follows from the fact that in polynomial time one can check whether or not the transition graph of a DFA matches a fixed pattern. The lower bound is obtained by a reduction to TL satisfiability.

The proof of Theorem 4 makes use of the notion of a stutter-free string, which is defined as follows. A string u is *stutter-free* if $u(i) \neq u(i+1)$ for all $i < |u|-1$. Clearly, every equivalence class of $\equiv_{st}$ contains exactly one stutter-free string. As a consequence of Lamport's observation, we note:

Lemma 5. *Let L be a stutter-invariant language over some alphabet Σ and $\varphi \in \mathrm{TL}[\mathsf{F}_{\mathsf{sf}}, \mathsf{U}_{\mathsf{sf}}]$ a formula over Σ such that $u \models \varphi$ iff $u \in L$, for $u \in \Sigma^+$ stutter-free. Then φ defines L.* □

So Theorem 4 will follow once we have established the following lemma.

Lemma 6. *Let $\boldsymbol{F}$ and $\boldsymbol{F}'$ be as in Theorem 4, and assume $\varphi \in \boldsymbol{F}'$. Then there exists $\varphi' \in \boldsymbol{F}$ such that $u \models \varphi$ iff $u \models \varphi'$, for $u \in \Sigma^+$ stutter-free.*

Proof. The proof is an inductive definition of φ', which works in both situations. The base case is trivial. In the induction step, negation and disjunction can be dealt with easily. What remains are formulas whose outermost connective is F or U. We set

$$\varphi' = \begin{cases} \displaystyle\bigvee_{a,b\in\Sigma:\ a\neq b} (a \wedge \mathsf{F}_{\mathsf{sf}}(b \wedge \mathsf{F}_{\mathsf{sf}}\psi')) \ , & \text{for } \varphi = \mathsf{F}\psi, \\ \displaystyle\bigvee_{a,b\in\Sigma:\ a\neq b} (a \wedge (a \,\mathsf{U}_{\mathsf{sf}}\, (b \wedge (\psi' \,\mathsf{U}_{\mathsf{sf}}\, \chi')))) \ , & \text{for } \varphi = \psi \,\mathsf{U}\, \chi. \end{cases}$$

We prove only that the second choice is correct; the proof that the first choice is correct is even simpler. First, assume $u \models \varphi$. Then there exists $i > 0$ such that $u(i,*) \models \chi$ and $u(j,*) \models \psi$ for $j \in \{1,\dots,i-1\}$. By induction hypothesis, this means $u(i,*) \models \chi'$ and $u(j,*) \models \psi'$ for $j \in \{1,\dots,i-1\}$. Clearly, we have $u \models u(0) \wedge u(0) \,\mathsf{U}_{\mathsf{sf}}\, (u(1) \wedge \psi' \,\mathsf{U}_{\mathsf{sf}}\, \chi')$, which is a disjunct of φ'.

Second, assume $u \models \varphi'$ and let a and b be symbols for which the corresponding disjunct holds. If $u \models a \wedge a \,\mathsf{U}_{\mathsf{sf}}\, (b \wedge \psi' \,\mathsf{U}_{\mathsf{sf}}\, \chi')$, then $u(0) = a$ and $u(1) = b$, since u is assumed to be stutter-free. But then $u(1,*) \models \psi' \,\mathsf{U}_{\mathsf{sf}}\, \chi'$, which implies, by induction hypothesis, $u(1,*) \models \psi \,\mathsf{U}_{\mathsf{sf}}\, \chi$, which, in turn, implies $u \models \psi \,\mathsf{U}\, \chi$. □

This completes the first part of this survey. We have seen how every fragment (determined by which modalities are allowed in forming formulas) of future temporal logic can be characterized in an effective, concise way by describing structural properties of DFA's.

5 Past Modalities and Symmetric Fragments

Thus far, we have only dealt with temporal modalities that refer to the future (and possibly the present) only. But, of course, each of the modalities considered has a symmetric past counterpart: S (since) goes with U, P (eventually in the past) goes with F, Y (previously) goes with X.

Adding past modalities does not increase the expressive power of temporal logic, i. e., $\mathrm{TL} = \mathrm{TL}[\mathsf{U}, \mathsf{S}]$. This is easy to see because for every temporal formula (with future and past modalities) one can still find a counter-free DFA recognizing the language defined by the formula. Similarly, $\mathrm{TL}[\mathsf{U}_{\mathsf{sf}}] = \mathrm{TL}[\mathsf{U}_{\mathsf{sf}}, \mathsf{S}_{\mathsf{sf}}]$, because even with past stutter-invariant modalities one can only express stutter-invariant languages. Clearly, $\mathrm{TL}[\mathsf{X}] = \mathrm{TL}[\mathsf{X}, \mathsf{Y}]$. But the expressive power of any other fragment is increased by adding the corresponding past modalities. Nevertheless, we have:

Theorem 5 (Decidability of Symmetric Fragments [16]). *For each of the fragments* $\mathrm{TL}[\mathsf{F}_{\mathsf{sf}}, \mathsf{P}_{\mathsf{sf}}]$*,* $\mathrm{TL}[\mathsf{F}, \mathsf{P}]$*, and* $\mathrm{TL}[\mathsf{X}, \mathsf{Y}, \mathsf{F}, \mathsf{P}]$ *it is decidable whether or not a given temporal property can be expressed in it.*

This theorem is based on similar structural characterizations as the ones given in Theorem 2 for the future fragments of temporal logic. There is, however, a fundamental difference. Instead of looking at the minimal DFA for a given language, one considers its syntactic semigroup, which, by definition, is symmetric in the sense that the syntactic semigroup of the reverse of a language is the reverse of the syntactic semigroup of the language, and thus better suited for investigating symmetric fragments. The proofs get more involved and require non-trivial finite semigroup theory. On the other hand, they also reveal interesting connections to first-order logic.

Remember that Kamp's theorem says that temporal logic (with future modalities only or with both) is as expressive as first-order logic. In this statement, a string $u \in \Sigma^+$ of length n is viewed as a structure in the signature with a binary predicate $<$, for the order relation on the positions, and unary predicates P_a, for each alphabet symbol a a separate predicate.

A simple induction shows that every temporal formula is equivalent to a first-order formula, even to a first-order formula that uses at most three variables. In view of Kamp's theorem, this means that temporal logic and first-order logic with three variables have the same expressive power. Reducing the number of variables to two leads to $\mathrm{TL}[\mathsf{F}, \mathsf{P}]$ and $\mathrm{TL}[\mathsf{X}, \mathsf{Y}, \mathsf{F}, \mathsf{P}]$, respectively:

Theorem 6 (Kamp's Theorem for Smaller Fragments [16]).
1. A language is expressible in $\mathrm{TL}[\mathsf{F}, \mathsf{P}]$ *if and only if it is expressible in first-order logic with two variables.*
2. A language is expressible in $\mathrm{TL}[\mathsf{X}, \mathsf{Y}, \mathsf{F}, \mathsf{P}]$ *if and only if it is expressible in first-order logic with two variables when the signature is extended by the built-in predicate* suc *for successor.*

There are more connections to first-order logic and to formal language theory. First, the languages expressible in $\mathrm{TL}[\mathsf{F}, \mathsf{P}]$ are exactly the unambiguous languages in the sense of Schützenberger [13]. Second, the languages expressible in $\mathrm{TL}[\mathsf{F}, \mathsf{P}]$ and $\mathrm{TL}[\mathsf{X}, \mathsf{Y}, \mathsf{F}, \mathsf{P}]$ are exactly the languages expressible by a Σ_2 and, at same time, a Π_2 formula (over the respective signature). For details, see [16].

6 Until Hierarchy

Which temporal modalities are needed to express a given temporal property is the first question to ask when one is interested in studying the expressive power of the temporal modalities themselves, but there are other, equally important ones, and some prominent ones are concerned with the "until hierarchy" of future temporal logic. The "until" modality is special in several respects. First, it is complete in the sense that no other modality is needed to express all temporal properties. Second, it is the only binary modality. The last fact is crucial; it makes formulas hard to read, especially, when nesting occurs. So the question is whether or not nesting of "until" is necessary, even when the other modalities can be used for free.[4] Using an appropriate Ehrenfeucht-Fraïssé game with additional types of rounds corresponding to X and U, one can actually show that the more nesting is allowed, the more one can express:

Theorem 7 (Strictness of Until Hierarchy [3]). *Let $\Sigma = \{a, b, c\}$ be a three-element alphabet and define φ_n, $n \geq 0$, by $\varphi_0 = a$ and $\varphi_{n+1} = a \wedge \mathsf{X}(b \mathbin{\mathsf{U}} \varphi_n)$. Then $\mathsf{F}\varphi_n$ is of until nesting depth n, but $\mathcal{L}_\Sigma(\varphi_n)$ is not definable by a formula of until nesting depth $< n$.*

We even have:

Theorem 8 (Computability of Until Depth [14]). *Given a temporal formula φ, one can compute the minimal until nesting depth required to express the language defined by φ.*

The proof of this theorem, just as the proof of Theorem 5, makes heavy use of finite semigroup theory. A key ingredient of the proof is the so-called semidirect product/substitution principle, which, in rough outline, states that if two fragments of temporal logic, say $\boldsymbol{F}$ and $\boldsymbol{G}$, are characterized by classes $\boldsymbol{V}$ and $\boldsymbol{W}$ of finite semigroups, then the fragment which is obtained by substituting formulas of $\boldsymbol{G}$ into formulas of $\boldsymbol{F}$ is characterized by the semidirect product of $\boldsymbol{V}$ and $\boldsymbol{W}$. Applied to the until hierarchy, this principle says that the k-th level of the hierarchy is characterized by a k-th power of the class of semigroups that characterizes level 1. (Observe that a formula of until depth at most k can be written as a k-fold substitution of formulas of depth at most 1, and vice versa.) For details, see [14] or [15].

7 Conclusion

The results presented in this survey show that there are intimate connections between temporal logic, the theory of finite automata, the theory of finite semigroups, and first-order logic. The classification of discrete temporal properties has been accomplished to a great extent. A problem that is still open is the decidability of the combined until/since hierarchy, where a property is classified according to the nesting depth in U and S required to express it using future as well as past modalities. Note that this hierarchy is known to be strict, see [3].

[4] In the literature, other binary modalities (such as "at next" [6] or "as long as" [7]) have been occasionally used, and these operators are as powerful as "until." In fact, nesting depth with regard to any of these two operators is exactly the same as nesting depth with respect to "until."

References

1. Joëlle Cohen, Dominique Perrin, and Jean-Eric Pin. On the expressive power of temporal logic. *J. Comput. System Sci.*, 46(3):271–294, 1993.
2. Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. In *Proceedings 12th Annual IEEE Symposium on Logic in Computer Science*, pages 228–235, Warsaw, Poland, 1997.
3. Kousha Etessami and Thomas Wilke. An until hierarchy for temporal logic. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 108–117, New Brunswick, N. J., 1996.
4. Dov M. Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal analysis of fairness. In *Conference Record of the 12th ACM Symposium on Principles of Programming Languages*, pages 163–173, Las Vegas, Nev., 1980.
5. Johan Anthony Willem Kamp. *Tense Logic and the Theory of Linear Order.* PhD thesis, University of California, Los Angeles, Calif., 1968.
6. Fred Kröger. *Temporal Logic of Programs.* Springer, Berlin, 1987.
7. Leslie Lamport. Specifying concurrent program modules. *ACM Trans. Programming Lang. Sys.*, 5(2):190–222, 1983.
8. Oded Maler. *Finite Automata: Infinite Behavior, Learnability and Decomposition.* PhD thesis, The Weizmann Institute of Science, Rehovot, Israel, 1990.
9. Oded Maler and Amir Pnueli. Tight bounds on the complexity of cascaded decomposition of automata. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, vol. II, pages 672–682, St. Louis, Miss., 1990.
10. Robert McNaughton and Seymour Papert. *Counter-Free Automata.* MIT Press, Cambridge, Mass., 1971.
11. Doron Peled and Thomas Wilke. Stutter-invariant temporal properties are expressible without the next-time operator. *Inform. Process. Lett.*, 63(5):243–246, 1997.
12. Marcel P. Schützenberger. On finite monoids having only trivial subgroups. *Inform. and Computation*, 8:190–194, 1965.
13. Marcel P. Schützenberger. Sur le produit de concatenation non ambigu. *Semigroup Forum*, 13:47–75, 1976.
14. Denis Thérien and Thomas Wilke. Temporal logic and semidirect products: An effective characterization of the until hierarchy. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 256–263, Burlington, Vermont, 1996.
15. Denis Thérien and Thomas Wilke. Temporal logic and semidirect products: An effective characterization of the until hierarchy. Technical report 96-28, DIMACS, Piscataway, N. J., 1996.
16. Denis Thérien and Thomas Wilke. Over words, two variables are as powerful as one quantifier alternation: $\mathrm{FO}^2 = \Sigma_2 \cap \Pi_2$. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 41–47, Dallas, Texas, 1998.
17. Thomas Wilke. Classifying discrete temporal properties. Habilitationsschrift (postdoctoral thesis), Kiel, Germany, 1998.