Relating Hierarchy of Temporal Properties to Model Checking

Ivana Černá and Radek Pelánek *

Department of Computer Science, Faculty of Informatics
Masaryk University Brno, Czech Republic
{cerna,xpelanek}@fi.muni.cz

Abstract. The hierarchy of properties as overviewed by Manna and Pnueli [18] relates language, topology, ω-automata, and linear temporal logic classifications of properties. We provide new characterisations of this hierarchy in terms of automata with Büchi, co-Büchi, and Streett acceptance condition and in terms of Σ_i^{LTL} and Π_i^{LTL} hierarchies. Afterwards, we analyse the complexity of the model checking problem for particular classes of the hierarchy and thanks to the new characterisations we identify those linear time temporal properties for which the model checking problem can be solved more efficiently than in the general case.

1 Introduction

Model checking has become a popular technique for formal verification of reactive systems. The model checking process has several phases – the major ones being modelling of the system, specification of desired properties of the system, and the actual process of automatic verification. Each of these phases has its specific difficulties. In this paper we study linear temporal properties and algorithms for the automatic verification of these properties.

Reactive systems maintain an ongoing interaction with their environment and thus produce computations – infinite sequences of states. When analysing the behaviour of such a system we are interested in some finite set AP of observable propositions about states. Hence, we can view a computation of the system as an infinite word over 2^{AP}. In general, we define a temporal *property* as a language of infinite words. A reactive system S is said to have a property P if all possible computations of S belong to P.

The problem of proper and correct specification of properties the system ought to satisfy led to a careful study of theoretical aspects of properties. Manna and Pnueli [18] have proposed a classification of temporal properties into a hierarchy. They characterise the classes of the hierarchy through four views: a language-theoretic view, a topological view, a temporal logic view, and an automata view. The fact that the hierarchy can be defined in many different ways shows the robustness of this hierarchy.

* Supported by GA ČR grant no. 201/03/0509

Model checking theory is devoted to the development of efficient algorithms for the automatic verification of properties of reactive systems. A very successful approach to verifying properties expressed as linear temporal logic (LTL) formulas makes use of automata over infinite words. Here the problem of verifying a property reduces to the problem whether a given automaton recognises a non-empty language (so called non-emptiness check). The complexity of the non-emptiness check depends on the type of the automaton. Bloem, Ravi, and Somenzi [1] have studied two specialised types of automata, called *weak* and *terminal*, for which the non-emptiness check can be performed more efficiently than in the general case.

Our contribution Our aim is to classify temporal properties specifiable by linear temporal logic formulas with respect to the complexity of their verification. To this end we provide a classification of temporal properties through two new views. First, we characterise properties in terms of automata over infinite words (ω-automata) with Büchi, co-Büchi, and Streett acceptance condition and in terms of weak and terminal automata. Weak and terminal automata are used in the verification process and are checked for non-emptiness.

For the second characterisation we introduce a new hierarchy (called Until-Release hierarchy) of LTL formulas based on alternation depth of temporal operators Until and Release. We provide a relationship between the Until-Release hierarchy and the hierarchy by Manna and Pnueli [18].

Our new classification provides us with an exact relationship between the type of a formula and the type of an automaton, which is checked for non-emptiness in the model checking process of the formula.

In the second part of the paper we enquire into particular automata and analyse the complexity of their non-emptiness check in connection with both explicit and implicit representation of automata. This gives us an exact relationship between types of properties and the complexity of their verification. Finally, we discuss the possibility of exact determination of the type of a formula.

Due to space limitations complete proofs (and some formal definitions) are omitted and can be found in the full version of the paper [3].

Related work The previous work on verification, which takes into account a classification of properties, is partly devoted to the proof-based approach to verification [4]. Papers on specialised model checking algorithms either cover only part of the hierarchy or have a heuristic nature. Vardi and Kupferman [15] study the model checking of safety properties. Schneider [19] is concerned with a translation of persistence properties into weak automata. Bloem and Somenzi study heuristics for the translation of a formula into weak (terminal) automaton [21] and suggest specialised algorithms for the non-emptiness problem [1]. Our work covers all types of properties and brings out the correspondence between the type and the complexity of non-emptiness check.

2 Hierarchy of Temporal Properties

The hierarchy studied by Manna and Pnueli [18] classifies properties into six classes: *guarantee, safety, obligation, persistence, recurrence,* and *reactivity* properties.

Definition 1 (Language-theoretic view [18]).
Let $P \subseteq \Sigma^\omega$ be a property over Σ.

- *P is a* safety *property if there exists a language of finite words $L \subseteq \Sigma^*$ such that for every $w \in P$ all finite prefixes of w belong to L.*
- *P is a* guarantee *property if there exists a language of finite words $L \subseteq \Sigma^*$ such that for every $w \in P$ there exists a finite prefix of w which belongs to L.*
- *P is an* obligation *property if P can be expressed as a positive boolean combination of safety and guarantee properties.*
- *P is a* recurrence *property if there exists a language of finite words $L \subseteq \Sigma^*$ such that for every $w \in P$ infinitely many prefixes of w belong to L.*
- *P is a* persistence *property if there exists a language of finite words $L \subseteq \Sigma^*$ such that for every $w \in P$ all but finitely many prefixes of w belong to L.*
- *P is a* reactivity *property if P can be expressed as a positive boolean combination of recurrence and persistence properties.*

In what follows, the abbreviation κ-property stands for a property of one of the six above mentioned types. Inclusions, which relate the corresponding classes into a hierarchy, are depicted in Fig. 1. Classes which are higher up strictly contain classes which are lower down.

2.1 Automata View

Manna and Pnueli [18] have defined the hierarchy of properties in terms of deterministic Streett predicate automata. Automata for considered classes of properties differ in restrictions on their transition functions and acceptance conditions. In this section we provide a new characterisation of the hierarchy in terms of deterministic ω-automata which uses only restrictions on acceptance conditions (the transition function is always the same). We find this characterisation more uniform and believe that it provides better insight into the hierarchy. On top of that we study other widely used types of ω-automata and show that each of them exactly corresponds to one class in the hierarchy.

An *ω-automaton* is a tuple $A = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, where Σ is a finite alphabet, Q is a finite set of states, $q_0 \in Q$ is an initial state, δ is a transition function, and α is an acceptance condition. The transition function determines four types of automata: *deterministic*, *nondeterministic*, *universal*, and *alternating*. A nondeterministic automaton has a transition function of the type $\delta : Q \times \Sigma \to 2^Q$. A run π of such an automaton on an infinite word $w = w(0)w(1)\ldots$ over Σ is a sequence of states $\pi = r_0, r_1, \ldots$ such that $r_0 = q_0$ and $r_{i+1} \in \delta(r_i, w(i))$ for each $i \geq 0$. A nondeterministic automaton accepts a word w if there exists an accepting run (see below) on w. Universal automata are defined in the same way, the only difference is that the universal automaton accepts a word w if all runs on w are accepting. Deterministic automata are such that $|\delta(q, a)| = 1$ for all $q \in Q, a \in \Sigma$ (there is a unique run on each word). Alternating automata form a generalisation of nondeterministic and universal automata.

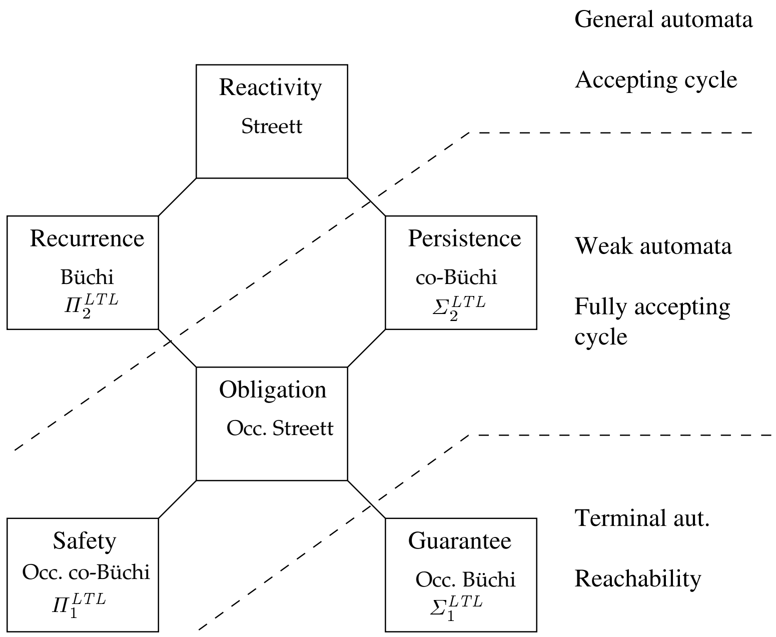

Fig. 1. Relations between classes of the hierarchy and their different characterisations. Classes which are higher up properly contain classes which are lower down. Classes on the same level are dual with respect to complementation, while the classes obligation and reactivity can be obtained by boolean combinations of properties from classes lower down

For a run π we define the *infinity set*, $\mathit{Inf}(\pi)$, to be the set of all states that appear infinitely often in π and the *occurrence set*, $\mathit{Occ}(\pi)$, to be the set of states that appear at least once in π. Acceptance conditions α are defined with respect to infinity set as follows:

- *Büchi* condition $\alpha \subseteq Q$: a run π is accepting iff $\mathit{Inf}(\pi) \cap \alpha \neq \emptyset$
- *co-Büchi* condition $\alpha \subseteq Q$: a run π is accepting iff $\mathit{Inf}(\pi) \cap \alpha = \emptyset$
- *Streett* condition $\alpha = \{\langle G_1, R_1\rangle, \ldots, \langle G_n, R_n\rangle\}, G_i, R_i \subseteq Q$: a run π is accepting iff $\forall i : (\mathit{Inf}(\pi) \cap G_i \neq \emptyset \Rightarrow \mathit{Inf}(\pi) \cap R_i \neq \emptyset)$

For every acceptance condition we can define its ”occurrence” version [17, 23] if $\mathit{Occ}(\pi)$ substitues for $\mathit{Inf}(\pi)$ (also called *Staiger-Wagner* acceptance). According to the acceptance condition we denote ω-automata as Büchi and occurence Büchi automata respectively and so on. A property P is defined to be

Table 1. The expressivity – each of 24 possible inter-combinations of the transition function and acceptance condition corresponds to one of the six hierarchy classes

				Occurrence		
	Büchi	co-Büchi	Streett	Büchi	co-Büchi	Streett
Deterministic	*recurrence*	*persistence*	*reactivity*	*guarantee*	*safety*	*obligation*
Nondeterministic	*reactivity*	*persistence*	*reactivity*	*persistence*	*safety*	*persistence*
Universal	*recurrence*	*reactivity*	*reactivity*	*guarantee*	*recurrence*	*recurrence*
Alternating	*reactivity*	*reactivity*	*reactivity*	*persistence*	*recurrence*	*reactivity*

specifiable by automata if there is an ω-automaton A which accepts a word w if and only if $w \in P$.

Theorem 1. *Let P be a property specifiable by automata. Then P is a guarantee, safety, obligation, persistence, recurrence, or reactivity property if and only if it is specifiable by a deterministic occurrence Büchi, occurrence co-Büchi, occurrence Streett, co-Büchi, Büchi, or Streett automaton respectively (see Table 1).*

Proof. For each class from the hierarchy a κ-automaton is defined in [18] by posing specific restrictions on transition functions and acceptance conditions of deterministic Streett predicate automata. Using an adjustment of accepting conditions and a copy construction one can effectively transform κ-automata to above mentioned automaton and vice versa. (Details can be found in the full version of the paper [3].) □

To make the picture complete we have examined other types of automata as well (see Table 1). For every possible combination of transition function and acceptance condition the class of specifiable properties exactly coincides with one class in the hierarchy. Results for infinite occurrence acceptance conditions follow from [16]. Universal occurrence Büchi and nondeterministic occurrence co-Büchi automata can be determinised through the power set construction and thus they recognise the same classes as their deterministic counterparts. The other results for occurrence acceptance condition follow from [17].

2.2 Linear Temporal Logic View

In this section we characterise the hierarchy of properties through a new hierarchy of LTL formulas based on an alternation depth.

The set of LTL formulas is defined inductively starting from a countable set AP of atomic propositions, Boolean operators, and the temporal operators **X** (Next), **U** (Until) and **R** (Release):

$$\Psi := a \mid \neg\Psi \mid \Psi \vee \Psi \mid \Psi \wedge \Psi \mid \mathbf{X}\Psi \mid \Psi\,\mathbf{U}\,\Psi \mid \Psi\,\mathbf{R}\,\Psi$$

LTL formulas are interpreted in the standard way on infinite words over the alphabet 2^{AP}. A property P is defined to be *specifiable by LTL* if there is an LTL formula φ such that $w \models \varphi$ if and only if $w \in P$.

In recent years, considerable effort has been devoted to the study of LTL hierarchies which were defined with respect to the number of nested temporal operators Until, Since, and Next ([10, 22, 14]). These hierarchies provide interesting characterizations of LTL definable languages. However, they do not seem to have a direct connection to the model checking problem. We propose a new hierarchy which is based on alternation depth instead of nested depth, and establish its connection with the hierarchy of properties. In the next Section we demonstrate that this classification directly reflects the hardness of the verification problem for particular properties.

Let us define hierarchies Σ_i^{LTL} and Π_i^{LTL}, which reflect alternations of Until and Release operators in formulas. We use the Σ/Π notation since the way the hierarchy is defined strongly resembles the quantifier alternation hierarchy of first-order logic formulas or fixpoints alternation hierarchy of μ-calculus formulas.

Definition 2.

The class $\Sigma_0^{LTL} = \Pi_0^{LTL}$ is the least set containing all atomic propositions and closed under the application of boolean and Next operators.
The class Σ_{i+1}^{LTL} is the least set containing Π_i^{LTL} and closed under the application of conjunction, disjunction, Next and Until operators.
The class Π_{i+1}^{LTL} is the least set containing Σ_i^{LTL} and closed under the application of conjunction, disjunction, Next and Release operators.

The following theorem shows that the type of a property and alternation depth of its specification are closely related.

Theorem 2. *A property that is specifiable by LTL is a guarantee (safety, persistence, recurrence respectively) property if and only if it is specifiable by a formula from the class Σ_1^{LTL} (Π_1^{LTL}, Σ_2^{LTL}, Π_2^{LTL} respectively) (see Fig. 1).*

Proof. The proof makes use of the classification of LTL formulas by Chang, Manna, and Pnueli [4]. Here every κ-property is syntactically characterised with the help of a κ-formula. We can transform any guarantee (safety, persistence, recurrence respectively) formula into an equivalent Σ_1^{LTL} (Π_1^{LTL}, Σ_2^{LTL}, Π_2^{LTL} respectively) formula.

Theorem 3. *A property is specifiable by LTL if and only if it is specifiable by a positive boolean combination of Σ_2^{LTL} and Π_2^{LTL} formulas. Therefore both Σ_i^{LTL} and Π_i^{LTL} hierarchies collapse in the sense that every LTL formula is specifiable both by a Σ_3^{LTL} and Π_3^{LTL} formula.*

3 Model Checking and Hierarchy of Properties

The model checking problem is to determine for a given reactive system K and a temporal formula φ whether the system satisfies the formula. A common approach to model checking of finite state systems and LTL formulas is to construct an automaton $A_{\neg\varphi}$ for the negation of the property and to model the system as an automaton K. The product automaton $K \times A_{\neg\varphi}$ is then checked for non-emptiness. The product automaton is a nondeterministic Büchi automaton. For the formal definition of the problem and detailed description of the algorithm we refer to [5].

Our aim is to analyse the complexity of the non-emptiness check depending on the type of the verified property. As the complexity of the non-emptiness check is determined by attributes of an automaton, the question is whether for different types of formulas one can construct different types of automata. We give a comprehensive answer to this question in this section. In the next section we demonstrate how the complexity of the non-emptiness check varies depending on the type of automata.

To classify nondeterministic Büchi automata we adopt the criteria proposed by Bloem, Ravi, and Somenzi [1]. They differentiate *general*, *weak*, and *terminal* automata according to the following restrictions posed on their transition functions:

- *general*: no restrictions
- *weak*: there exists a partition of the set Q into components Q_i and an ordering $\leq$ on these sets, such that for each $q \in Q_i, p \in Q_j$, if $\exists a \in \Sigma : q \in \delta(p, a)$ then $Q_i \leq Q_j$. Moreover for each Q_i, $Q_i \cap \alpha = \emptyset$, in which case Q_i is a rejecting component, or $Q_i \subseteq \alpha$, in which case Q_i is an accepting component.
- *terminal*: for each $q \in \alpha, a \in \Sigma$ it holds $\delta(q, a) \neq \emptyset$ and $\delta(q, a) \subseteq \alpha$.

Each transition of a weak automaton leads to a state in either the same or lower component. Consequently each run of a weak automaton gets eventually trapped within one component. The run is accepting iff this component is accepting. The transition function of a terminal automaton is even more restricted – once a run of a terminal automaton reaches an accepting state the run is accepting regardless of the suffix. Terminal and weak automata are jointly called *specialised* automata. It shows up that the classes of properties specifiable by weak and terminal automata coincide with classes of the hierarchy.

Theorem 4. *A property P specifiable by automata is a guarantee (persistence) property if and only if it is specifiable by a terminal (weak) automaton.*

Theorem 4 raises a natural question whether and how effectively one can construct for a given guarantee (persistence) formula the corresponding terminal (weak) automaton. A construction of an automaton for an LTL formula was first proposed by Wolper, Vardi, and Sistla [24]. This basic construction has been improved in several papers ([12, 21, 9]) where various heuristics have been used to produce automaton as small and as "weak" as possible. Although these heuristics are quite sophisticated, they do not provide any insight into the relation between the formula and the "weakness" of the resulting automaton. Constructions for special types of properties can be found in [19, 15].

We present a new modification of the original construction which yields for a formula from the class Σ_1^{LTL} and Σ_2^{LTL} a specialised automaton.

Theorem 5. *For every Σ_1^{LTL} (Σ_2^{LTL}) formula φ we can construct a terminal (weak) automaton accepting the property defined by φ.*

Proof. States of the automaton are sets of subformulas of the formula φ. The transition function is constructed in such a way that the following invariant is valid: if the automaton is in a state S then the remaining suffix of the word should satisfy all formulas in S. The acceptance condition is used to enforce the fulfillment of Until operators. For Σ_1^{LTL} and Σ_2^{LTL} formulas the acceptance condition can be simplified thanks to the special structure of alternation of Until and Release operators in the formula. □

4 Non-Emptiness Algorithms

In the previous section we showed that we can effectively construct specialised automata for formulas from lower classes of the hierarchy. Since the verified system K can be modelled as an automaton without acceptance conditions, the type of the product automaton is determined entirely by the type of the automaton $A_{\neg\varphi}$, that is even the product automaton is specialised. In this section we revise both explicit and symbolic non-emptiness algorithms for different types of automata.

General Automata

For general automata the non-emptiness check is equivalent to the reachability of an accepting cycle (i.e. cycle cointaining an accepting state). The most efficient explicit algorithm is the nested depth-first search (DFS) algorithm [6, 13]. With the symbolic representation one has to use nested fixpoint computation (e.g. Emerson-Lei algorithm) with a quadratic number of symbolic steps (for an overview of symbolic algorithms see [11]).

Weak Automata

States of a weak automaton are partitioned into components and therefore states from each cycle are either all accepting (the cycle is fully accepting) or all non-accepting. The non-emptiness problem is equivalent to the reachability of a fully accepting cycle. The explicit algorithm has to use only a single DFS [8] in this case. With the symbolic representation single fixpoint computation [1] with a linear number of steps is sufficient.

Terminal Automata

Once a terminal automaton reaches an accepting state, it accepts the whole word. Thus the non-emptiness of a terminal automaton can be decided by a simple reachability analysis.

With the symbolic representation there is even asymptotical difference between the algorithms for general and specilized cases. All explicit algorithms have linear time complexity, but the use of specialized algorithms still brings several benefits. Time and space optimalizations, “Guided search” heuristics [8], and the partial-order reduction [13] can be employed more directly for specialized algorithms. Algorithms for specialized automata can be more effectively transformed to distributed ones [2].

These benefits were already experimentally demonstrated. Edelkamp, Lafuente, and Leue [8] extended the explicit model checker SPIN by a non-emptiness algorithm which to a certain extent takes the type of an automaton into consideration. Bloem, Ravi, and Somenzi [1] performed experiments with symbolic algorithms and in [2] experiments with distributed algorithms are presented.

5 Conclusions

The paper provides a new classification of temporal properties through deterministic ω-automata and through the Until-Release hierarchy. It provides effective transformation of the Σ_1^{LTL} (Σ_2^{LTL}) formula into terminal (weak) automaton and it argues that the non-emptiness problem for these automata can be solved more efficiently.

It is decidable whether given formula specifies property of type κ [3]. In a case that it is guarantee (persistence) formula it is possible to transform it into an equivalent Σ_1^{LTL} (Σ_2^{LTL}) formula. Thus the new classifications provide us with exact relationship between the type of a formula and the type of the non-emptiness problem.

The determination of the type of a formula and the transformation are rather expensive (even deciding whether a given formula specifies a safety property is PSPACE-complete [20]). However, formulas are usually quite short and it is typical to make many tests for one fixed formula. In such a case, the work needed for determining the type of the formula is amortised over its verification.

Moreover, most of the practically used formulas are simple. We have studied the Specification Patterns System [7] that is a collection of the most often verified properties. It shows up that most of the properties can be easily transformed into terminal (41%) or weak (54%) automata. We conclude that model checkers should take the type of the property into account and use the specialized non-emptiness algoritms as often as possible.

References

1. R. Bloem, K. Ravi, and F. Somenzi. Efficient decision procedures for model checking of linear time logic properties. In *Proc. Computer Aided Verification*, volume 1633 of *LNCS*, pages 222–235. Springer, 1999.
2. I. Černá and R. Pelánek. Distributed explicit fair cycle detection. In *Proc. SPIN Workshop on Model Checking of Software*, volume 2648 of *LNCS*, pages 49–73. Springer, 2003.
3. I. Černá and R. Pelánek. Relating hierarchy of linear temporal properties to model checking. Technical Report FIMU-RS-2003-03, Faculty of Informatics, Masaryk University, 2003. http://www.fi.muni.cz/informatics/reports/.
4. E. Y. Chang, Z. Manna, and A. Pnueli. Characterization of temporal property classes. In *Proc. Automata, Languages and Programming*, volume 623 of *LNCS*, pages 474–486. Springer, 1992.

5. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999.
6. C. Courcoubetis, M. Vardi, P. Wolper, and M. Yannakakis. Memory-efficient algorithms for the verification of temporal properties. *Formal Methods in System Design*, 1:275–288, 1992.
7. M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Property specification patterns for finite-state verification. In *Proc. Workshop on Formal Methods in Software Practice*, pages 7–15. ACM Press, 1998.
8. S. Edelkamp, A. L. Lafuente, and S. Leue. Directed explicit model checking with HSF-SPIN. In *Proc. SPIN Workshop on Model Checking of Software*, volume 2057 of *LNCS*, pages 57–79. Springer, 2001.
9. K. Etessami and G. J. Holzmann. Optimizing Büchi automata. In *Proc. CONCUR*, volume 1877 of *LNCS*, pages 153–167. Springer, 2000.
10. K. Etessami and T. Wilke. An Until hierarchy for temporal logic. In *Proc. IEEE Symposium on Logic in Computer Science*, pages 108–117. Computer Society Press, 1996.
11. K. Fisler, R. Fraer, G. Kamhi Y. Vardi, and Zijiang Yang. Is there a best symbolic cycle-detection algorithm? In *Proc. Tools and Algorithms for Construction and Analysis of Systems*, volume 2031 of *LNCS*, pages 420–434. Springer, 2001.
12. R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Proc. Protocol Specification Testing and Verification*, pages 3–18. Chapman & Hall, 1995.
13. G. J. Holzmann, D. Peled, and M. Yannakakis. On nested depth first search. In *Proc. SPIN Workshop*, pages 23–32. American Mathematical Society, 1996.
14. A. Kučera and J. Strejček. The stuttering principle revisited: On the expressiveness of nested X and U operators in the logic LTL. In *Proc. Computer Science Logic*, volume 2471 of *LNCS*, pages 276–291. Springer, 2002.
15. O. Kupferman and M. Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001.
16. C. Löding. Methods for the transformation of omega-automata: Complexity and connection to second order logic. Master's thesis, Christian-Albrechts-University of Kiel, 1998.
17. C. Löding and W. Thomas. Alternating automata and logics over infinite words. In *Proc. IFIP International Conference on Theoretical Computer Science*, volume 1872 of *LNCS*, pages 521–535. Springer, 2000.
18. Z. Manna and A. Pnueli. A hierarchy of temporal properties. In *Proc. ACM Symposium on Principles of Distributed Computing*, pages 377–410. ACM Press, 1990.
19. K. Schneider. Improving automata generation for linear temporal logic by considering the automaton hierarchy. In *Proc. Logic for Programming, Artificial Intelligence, and Reasoning*, volume 2250 of *LNCS*, pages 39–54. Springer, 2001.
20. A. P. Sistla. Safety, liveness, and fairness in temporal logic. *Formal Aspects of Computing*, 6(5):495–512, 1994.
21. F. Somenzi and R. Bloem. Efficient Büchi automata from LTL formulae. In *Proc. Computer Aided Verification*, volume 1855 of *LNCS*, pages 248–263. Springer, 2000.
22. D. Therien and T. Wilke. Nesting Until and Since in linear temporal logic. In *Proc. Symposium on Theoretical Aspects of Computer Science*, volume 2285 of *LNCS*, pages 455–464. Springer, 2002.
23. W. Thomas. Languages, automata and logic. In *Handbook of Formal Languages*, volume 3, pages 389–455. Springer, 1997.
24. P. Wolper, M.Y. Vardi, and A.P. Sistla. Reasoning about inifinite computation paths. In *Proc. Symp. on Foundations of Computer Science*, pages 185 – 194, Tuscon, 1983.