A Unified Approach For Showing Language Containment and Equivalence Between Various Types of ω-Automata

E.M. Clarke   I.A. Draghicescu   R.P. Kurshan[1]

September 1989
CMU-CS-89-192 2

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[1]AT&T Bell Laboratories, Murray Hill, NJ

ABSTRACT

We consider the language containment and equivalence problems for six different types of ω-automata: Büchi, Muller, Rabin, Streett, the L-automata of Kurshan, and the ∀ automata of Manna and Pnueli. We give a six by six matrix in which each row and column is associated with one of these types of automata. The entry in the i^{th} column is the complexity of showing containment between the i^{th} type of automation and j^{th}. Thus, for example, we give the complexity of showing language containment and equivalence between a Büchi automaton and a Muller or Streett automaton. Our results are obtained by a uniform method that associates a formula of the computation tree logic CTL* with each type of automaton. Our algorithms use a *model checking* procedure for the logic with the formulas obtained from the automata. The results of our paper are important for verification of finite state concurrent systems with fairness constraints. A natural way of reasoning about such systems is to model the finite state program by one ω-automaton and its specification by another.

This research was partially supported by National Science Foundation grant CCR-87-226-33. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation or the U.S. Government.

1. Introduction

1.1. Background

ω–Automata were first used by Büchi in a paper on the decision problem for the logic S1S [7]. A short time later Muller showed that such automata were also useful for modeling the behavior of asynchronous circuits [11]. Like a conventional automaton on finite words, an ω–automaton consists of a set of states, an input alphabet, a transition relation and a start state. The difference between the two occurs in the definition of what it means for a word to be *accepted* by an automaton. Since the notion of a final state is not appropriate for a machine that accepts infinite words, another method must be used for defining acceptance. In Büchi's definition, some states were specified as *accepting states*. In order for a word to be accepted, these states must occur infinitely often during a run of the machine on the word. The definition that Muller used was somewhat more complicated. His acceptance condition consisted of set in which each element was a set of states. In order for a word to be accepted by an automaton, the set of states that occur infinitely often during a run of the machine on the word must be one of the elements of the acceptance set. Other acceptance conditions have been given by Rabin [13], Streett [16], Kurshan [9], and Manna and Pnueli [10]. It can be shown that each type of automaton accepts the same class of languages (i.e. the ω–regular langauges [1]); however, the translation from one type of automaton to another may be quite complex [14].

The language containment and equivalence problems for ω–automata are defined in exactly the same way as for automata on finite words. Let M_1 and M_2 be two automata on infinite words with the same alphabet Σ. $\mathcal{L}(M_i)$ will be the language accepted by M_i. The *language containment problem* (or simply the *containment* problem when this is unambiguous) is the problem of determining whether $\mathcal{L}(M_1) \subseteq \mathcal{L}(M_2)$. The *equivalence problem*, on the other hand, is the problem of deciding whether $\mathcal{L}(M_1) = \mathcal{L}(M_2)$. Given an algorithm for the containment problem, we can easily obtain an algorithm for the equivalence problem, since $\mathcal{L}(M_1) = \mathcal{L}(M_2)$ iff $\mathcal{L}(M_1) \subseteq \mathcal{L}(M_2)$ and $\mathcal{L}(M_2) \subseteq \mathcal{L}(M_1)$. If M_2 is nondeterministic, then determining whether $\mathcal{L}(M_1) \subseteq \mathcal{L}(M_2)$ will in general be PSPACE hard, since the corresponding problem for ordinary automata on finite words has this complexity [2]. Consequently, in this paper we will only consider the case in which M_2 is deterministic. M_1, however, can be either deterministic or nondeterministic.

In recent years the study of ω–automata has experienced a somewhat surprising rebirth. The renewed interest is apparently due to several factors. First of all, there has been a significant amount of research during this period on abstract models for concurrent programs. An important part of this research has been the study of various notions of *fairness*. Several of these notions involve some event holding infinitely often. Because of the similarity to the

[1]In some cases the automata must be nondeterministic to achieve this result. For example, deterministic Büchi automata are strictly less powerful than nondeterministic Büchi automata.

[2]This complexity is reversed in the case of $\forall$–automata. See Section 4.

way that acceptance is defined for Büchi automata, it is natural to use such automata in modeling programs with this type of fairness constraint. Some automatic verification techniques for finite state concurrent programs have exploited this similarity with considerable success. The approach used by Kurshan [9], models both the program and its specification by ω-automata. To show that a program is correct, he uses an algorithm for testing containment between the two such automata [8]. A second reason for interest in ω-automata comes from research on temporal logic. There is a close relationship between ω-automata and the models for a formula of linear temporal logic. Specifically, given a formula f of linear temporal logic, it is possible to construct a Büchi automaton that accepts those infinite sequences that are models for f. This relationship has also been exploited in an approach to automatic verification called temporal logic *model checking* ([2], [3]). In this case the specification of a finite state program is given by a temporal logic formula. By the property mentioned above it is possible to extract a Büchi automaton from the temporal logic formula and show containment in the same way that Kurshan does. Finally, research in VLSI on problems like clock skew has led to increased interest in asynchronous circuits. Models for such circuits like the one originally proposed by Muller have been resurrected in hopes of obtaining a better understanding for this class of circuits [4].

1.2. New Results of this paper

We consider the problem of deciding containment between *all* of the various types of ω-automata mentioned in the first paragraph. We give a 6×6 matrix where each row and column corresponds to one of the types of automata (See the figure at the end of Section 5.). The entry in the i^{th} row and j^{th} column is the complexity of showing containment between the i^{th} type of automata and j^{th}. The entries on the diagonal of the matrix give the complexity of deciding containment of two automata of the same type. We give a single uniform framework for establishing all of these results. We show how each entry in the matrix can be reduced to the problem of determining whether a certain temporal logic formula is true of a Kripke structure obtained from the two automata. We can efficiently determine whether the formula is true of the structure by using a model checking algorithm for the logic.

The particular logic that we use is called CTL* ([2], [3], [5]). It combines both branching-time and linear-time operators and is quite expressive. The syntax includes *path quantifiers*, A ("for all paths") and E ("for some path"), that are used as prefixes for formulas containing arbitrary combinations of the usual linear time operators G ("always"), F ("sometimes"), X ("nexttime"), and U ("until"). Although the model checking problem for full CTL* is PSPACE-complete [15], Emerson and Lei [6] give a restricted class of CTL* formulas (called *fair-CTL*) for which there is a model checking algorithm with polynomial complexity in the size of the CTL* formula and also in the size of the Kripke structure. We use a modification of this algorithm to obtain our results.

Our strategy for all of the cases in the matrix is essentially the same. We first express the acceptance conditions for the two automata by a formula in CTL*. Then we manipulate the formula to obtain one that can be handled by the model checking algorithm of Emerson and Lei. Since we are able to solve the containment problem by using an an efficient algorithm with practical complexity, the algorithms that we obtain for the entries in the matrix have practical complexity as well and are reasonably easy to implement. Moreover, since we use a uniform approach for obtaining our results, it is relatively simple to understand how the differences in the complexity among the various entries arise.

Although some of our results were previously known (See [8] for instance.), most of our results are new, because no one else has considered the hybrid cases (Büchi contained in Streett, etc.) that we consider. Even some of the cases on the diagonal are new. For example, we give a low order polynomial algorithm for deciding containment between deterministic Muller automata. As far as we know, no polynomial algorithm has been given for this case before. A naive algorithm to solve this problem would probably have exponential complexity.

1.3. Outline of paper

Our paper is organized as follows: In Section 2 we give formal definitions for the various types of ω–automata that we consider in this paper. In Section 3 we give the syntax and semantics for the branching-time temporal logic CTL*, and briefly discuss the model checking algorithm of Emerson and Lei. We precisely state the problem that the algorithm solves and give its complexity in the size of the CTL* formula and the size of the Kripke structure. Section 4 is the heart of the paper. In this section we show how to describe the various types of automata in CTL* and tell how to use the fair–CTL model checking algorithm for deciding containment between different types of machines. The paper concludes in Section 5 with a discussion of our results and some directions for future research.

2. ω-Automata

A *(nondeterministic)* ***ω-automaton*** over an *alphabet* Σ is a tuple (S, s_0, δ, F) where S is a finite set of *states*, s_0 is an *initial state*, $\delta : S \times \Sigma \rightarrow \mathcal{P}(S)$ is a *transition relation* and F is an *acceptance condition*. The automaton is *deterministic* if $\forall s \in S, \forall a \in \Sigma \ : \mid \delta(s,a) \mid \leq 1$. The automaton is ***complete*** if $\forall s \in S, \forall a \in \Sigma \ : \mid \delta(s,a) \mid \geq 1$. In this paper we will always assume that the automata are complete. It is easy to see that this does not affect the complexity of containment.

A *path* in M is an infinite sequence of states $s_0 s_1 s_2 \ldots \in S$ that starts in the initial state and has the property that $\forall i \geq 1, \exists a_i \in \Sigma \ : \ \delta(s_i, a_i) \ni s_{i+1}$. A path $s_0 s_1 s_2 \ldots \in S^\omega$ in M is a run of an infinite word $a_1 a_2 \ldots \in \Sigma^\omega$ if $\forall i \geq 1 \ : \ \delta(s_i, a_i) \ni s_{i+1}$.

An infinite word is accepted by a Büchi, Muller, Rabin, Streett or L automaton if it has an accepting run in the automaton. An infinite word is accepted by a $\forall$-automaton if all its possible runs in the automaton are accepted.

$\mathcal{L}(M) = \{a_1a_2\ldots \in \Sigma^\omega \mid a_1a_2\ldots \text{ is accepted by } M\}$.

The ***infinitary set*** of a sequence $s_0s_1s_2\ldots \in S^\omega$, $inf(s_0s_1\ldots)$, is the set of all the states that appear infinitely many times in the sequence.

If M is a Büchi automaton then $F \subseteq S$ is a set of states (as in the case of automata on finite words) and a run r is accepted by M if $inf(r) \cap F \neq \emptyset$.

The acceptance condition of a Muller automaton is a set $F \subseteq \mathcal{P}(S)$ of sets of states. A run is accepted by the Muller automaton if $inf(r) \in F$.

Lemma 1 *A run r is not accepted by a Muller automaton (S, s_0, δ, F) if and only if one of the following conditions holds :*

1. $\forall A \in F \,:\, inf(r) \not\subseteq A$ *or*

2. $\exists A \in F$, $\exists t \in A$ *such that:*

  (a) $\forall B \in F,\ B \subset A \,:\, inf(r) \not\subseteq B$ *and*

  (b) $inf(r) \subseteq A \setminus \{t\}$

Proof The "$\Rightarrow$" direction is proved by the following argument: Suppose that $inf(r) \notin F$. Then either $inf(r)$ is not contained in any set of F, and in this case 1 holds, or $inf(r)$ is contained in some set in F. Let A be a minimal set in F such that $inf(r) \subseteq A$. Then 2a holds and as $inf(r)$ must be strictly contained in A, there exists a $t \in A$ for which 2b holds. The "$\Leftarrow$" direction is equally simple. If case 1 holds then clearly $inf(r) \notin F$. Suppose 2 holds. As $inf(r)$ is strictly included in A and is not equal to any of the subsets of A that are in F it follows in this case also that $inf(r) \notin F$.

In the case of Rabin automata, the acceptance condition has the form $F = \{(U_1, V_1), \ldots, (U_n, V_n)\}$, where $U_i, V_i \subseteq S$. A run is accepted by M if there exists $i \in \{1, \ldots, n\}$ such that $inf(r) \subseteq U_i$ and $inf(r) \cap V_i \neq \emptyset$.

The Streett acceptance condition has the same form as that of Rabin, but the semantics is different. A run is accepted by a Streett automaton with $F = \{(U_i, V_i), \ldots, (U_n, V_n)\}$ if for every $i \in \{1, \ldots, n\}$, $inf(r) \subseteq U_i$ or $inf(r) \cap V_i \neq \emptyset$.

If M is an L automaton, the acceptance condition is a pair $F = (Z, V)$, where $Z \subseteq \mathcal{P}(S)$ and $V \subseteq S$. A run is accepted by the automaton if either $inf(r) \subseteq U$ for some $U \in Z$ or $inf(r) \cap V \neq \emptyset$.

The acceptance condition of a $\forall$-automaton is $F = (U, V) \subseteq S \times S$. A run is accepted by the automaton if either $inf(r) \subseteq U$ or $inf(r) \cap V \neq \emptyset$.

3. The Computation Tree Logic CTL*

There are two types of formulas in CTL*: *state formulas* (which are true in a specific state) and *path formulas* (which are true along a specific path). Let AP be the set of atomic proposition names. A state formula is either:

- A, if $A \in AP$.
- If f and g are state formulas, then $\neg f$ and $f \vee g$ are state formulas.
- If f is a path formula, then $\mathbf{E} f$ is a state formula.

A path formula is either:

- A state formula.
- If f and g are path formulas, then $\neg f$, $f \vee g$, $\mathbf{X} f$, and $f \mathbf{U} g$ are path formulas.

CTL* is the set of state formulas generated by the above rules.

We define the semantics of CTL* with respect to a structure $M = (\mathcal{S}, \mathcal{R}, \mathcal{L})$, where

- $\mathcal{S}$ is a set of states.
- $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ is the transition relation, which must be total. We write $s_1 \rightarrow s_2$ to indicate that $(s_1, s_2) \in \mathcal{R}$.
- $\mathcal{L} : \mathcal{S} \rightarrow \mathcal{P}(AP)$ is a function that labels each state with a set of atomic propositions true in that state.

Unless otherwise stated, all of our results apply only to *finite* Kripke structures.

We define a *path in M* to be a sequence of states, $\pi = s_0 s_1 \ldots$ such that for every $i \geq 0$, $s_i \rightarrow s_{i+1}$. π^i will denote the *suffix* of π starting at s_i.

We use the standard notation to indicate that a state formula f holds in a structure: $M, s \models f$ means that f holds at state s in structure M. Similarly, if f is a path formula, $M, \pi \models f$ means that f holds along path π in structure M. The relation $\models$ is defined inductively as follows (assuming that f_1 and f_2 are state formulas and g_1 and g_2 are path formulas):

1.	$s \models A$	*iff*	$A \in L(s)$.
2.	$s \models \neg f_1$	*iff*	$s \not\models f_1$.
3.	$s \models f_1 \vee f_2$	*iff*	$s \models f_1$ or $s \models f_2$.
4.	$s \models \mathbf{E}(g_1)$	*iff*	there exists a path π starting with s such that $\pi \models g_1$.
5.	$\pi \models f_1$	*iff*	s is the first state of π and $s \models f_1$.
6.	$\pi \models \neg g_1$	*iff*	$\pi \not\models g_1$.
7.	$\pi \models g_1 \vee g_2$	*iff*	$\pi \models g_1$ or $\pi \models g_2$.
8.	$\pi \models \mathbf{X} g_1$	*iff*	$\pi^1 \models g_1$.
9.	$\pi \models g_1 \mathbf{U} g_2$	*iff*	there exists a $k \geq 0$ such that $\pi^k \models g_2$ and for all $0 \leq j < k$, $\pi^j \models g_1$.

We will also use the following abbreviations in writing CTL* formulas:

- $f \wedge g \equiv \neg(\neg f \vee \neg g)$ • $\mathbf{A}(f) \equiv \neg\mathbf{E}(\neg f)$
- $\mathbf{F}f \equiv \mathit{true}\mathbf{U}f$ • $\mathbf{G}f \equiv \neg\mathbf{F}\neg f$.

Let $K = (\mathcal{S}, \mathcal{R}, \mathcal{L})$ be a finite Kripke structure. The *model checking problem* for a logic L is the problem of determining which states in $\mathcal{S}$ satisfy a given formula f of L. This problem is PSPACE–complete for CTL* [15]. However, for restricted CTL* formulas of the form

$$\mathbf{E}[\bigvee_{i=1}^{n}(\bigwedge_{j=1}^{n_i}(\mathbf{FG}p_{ij} \vee \mathbf{GF}q_{ij}))]$$

where p_{ij} and q_{ij} are propositional formulas, Emerson and Lei [6] give a polynomial model checking algorithm.

Theorem 1 *Let $K = (\mathcal{S}, \mathcal{R}, \mathcal{L})$ be a Kripke structure and f be a CTL* formula of the above form. There is an algorithm for finding the states of S where f is true that runs in time*

$$\mathcal{O}(\sum_{i=1}^{n} n_i \mid \mathcal{R} \mid + \sum_{i=1}^{n} n_i^2 \mid \mathcal{S} \mid + T)$$

where T is the time necessary to label the states satisfying p_{ij} and q_{ij} for $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, n_i\}$.

In this paper we will use a class of CTL* formulas of a somewhat more complicated although equivalent form in order to obtain tighter time bounds. The new formulas have the form

$$\mathbf{E}[\bigvee_{i=1}^{n}(\bigwedge_{j=1}^{n_i}(\mathbf{FG}p_{ij} \vee \mathbf{GF}q_{ij}) \wedge (\bigwedge_{j=1}^{m_i} \mathbf{GF}r_{ij}) \wedge \mathbf{FG}p_i)].$$

In this case the complexity of the model checking problem is

$$\mathcal{O}((\sum_{i=1}^{n}(n_i + 1) + 1) \mid \mathcal{R} \mid + \sum_{i=1}^{n}(n_i + 1)(n_i + m_i + 1) \mid \mathcal{S} \mid + T)$$

where T is the time required to find the sets of states satisfying p_{ij}, q_{ij}, r_{ij} and p_i for all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, n_i\}$. Although this complexity result is strictly tighter than the one mentioned in the previous theorem, its proof can be obtained by directly rewriting the proof of Emerson and Lei for this particular type of CTL* formula.

4. Complexity Results for Various Types of ω-automata

Our approach for computing the complexity of containment between various types of ω-automata is to transform these problems to model checking problems and then use Emerson and Lei's polynomial time algorithm. Alternatively, we could transform Emerson and Lei's algorithm to one that deals only with automata and avoid the use of temporal logic altogether. We decided to use the former approach because the containment problems can be expressed in temporal logic in a clear, succinct, and uniform manner. Moreover, by adopting this approach we could use an existing algorithm that has been efficiently implemented and thoroughly debugged.

4.1. Reducing the containment problem to model checking

Let $M = (S, s_0, \delta, F)$ be a Büchi, Muller, Rabin, Streett, L or $\forall$-automaton. Let ϕ_F be a linear formula over S that expresses the acceptance condition of M, more precisely, ϕ_F is a linear formula over S that has the property: an infinite path in M is accepted by F if and only if it satisfies ϕ_F. Let $\neg\phi_F$ be a linear formula over S that expresses the negation of the acceptance condition, i.e. has the property: an infinite word in S^ω satisfies $\neg\phi_F$ if and only if it does not satisfy ϕ_F. We will show in the next subsection that for any of the six types of ω-automata there are formulas ϕ_F and $\neg\phi_F$ of the form

$$\bigvee_{i=1}^{n}\Big(\bigwedge_{j=1}^{n_i}(\mathbf{FG}p_{ij} \vee \mathbf{GF}q_{ij}) \wedge \big(\bigwedge_{j=1}^{m_i}\mathbf{GF}r_{ij}\big) \wedge \mathbf{FG}p_i\Big)$$

where p_{ij}, q_{ij}, r_{ij} and p_i are propositional formulas.

We now describe how to compute the complexity of the containment problem by using the formulas for ϕ_F and $\neg\phi_F$. Let $M = (S, s_0, \delta, F)$ and $M' = (S', s'_0, \delta', F')$ be two complete Büchi, Muller, Rabin, Streett, L or $\forall$-automata over Σ such that $S \cap S' = \emptyset$.

Let $K(M, M') = (S \times S', (s_0, s'_0), \mathcal{L}, \mathcal{R})$ be the Kripke structure over $S \cup S'$ for which $\mathcal{L}(s, s') = \{s, s'\}$ and $(s, s')\mathcal{R}(t, t') \Leftrightarrow (\exists a \in \Sigma : \delta(s, a) \ni t$ and $\delta'(s', a) \ni t')$.

If M is a (nondeterministic) Büchi, Muller, Rabin, Streett, L or a deterministic $\forall$-automaton and M' is a deterministic Büchi, Muller, Rabin, Streett, L or a (nondeterministic) $\forall$-autmaton, then

$$\mathcal{L}(M) \subseteq \mathcal{L}(M') \Leftrightarrow K(M, M') \models \neg\mathbf{E}(\phi_F \wedge \neg\phi_{F'})$$

where ϕ_F expresses the accefptance condition of M and $\neg\phi_{F'}$ expresses the negation of the acceptance condition of M' . Note that the above equivalence holds if M accepts an infinite word if and only if there *exists* an accepting run in the automaton and if M' accepts an infinite word if an only if *all* its runs are accepting. (Since deterministic automata always have exactly one possible run, a deterministic automaton will satisfy the conditions for both M and M'.)

Suppose that ϕ_F and $\neg\phi_{F'}$ have the form :

$$\phi_F = \bigvee_{i=1}^{n}(\bigwedge_{j=1}^{n_i}(\mathbf{FG}p_{ij} \vee \mathbf{GF}q_{ij}) \wedge \bigwedge_{j=1}^{m_i}\mathbf{GF}r_{ij} \wedge \mathbf{FG}p_i)$$

$$\neg\phi_{F'} = \bigvee_{i=1}^{n'}(\bigwedge_{j=1}^{n'_i}(\mathbf{FG}p'_{ij} \vee \mathbf{GF}q'_{ij}) \wedge \bigwedge_{j=1}^{m'_i}\mathbf{GF}r'_{ij} \wedge \mathbf{FG}p'_i).$$

Then $K(M,M') \models \mathbf{E}(\phi_F \wedge \neg\phi_{F'})$ if and only if

$$K(M,M') \models \mathbf{E}[\bigvee_{i=1}^{n}\bigvee_{k=1}^{n'}(\bigwedge_{j=1}^{n_i}(\mathbf{FG}p_{ij}\vee\mathbf{GF}q_{ij})\wedge\bigwedge_{l=1}^{n'_k}(\mathbf{FG}p'_{kl}\vee\mathbf{GF}q'_{kl})\wedge\bigwedge_{j=1}^{m_i}\mathbf{GF}r_{ij}\wedge\bigwedge_{l=1}^{m'_k}\mathbf{GF}r'_{kl}\wedge\mathbf{FG}(p_i\wedge p'_k))]$$

and therefore, as shown in Section 3, the inclusion $\mathcal{L}(M) \subseteq \mathcal{L}(M')$ can be checked in time

$$\mathcal{O}(\sum_{i=1}^{n}\sum_{k=1}^{n'}(n_i + n'_k + 1)(\mid \delta \mid\mid \delta' \mid + \mid S \mid\mid S' \mid (n_i + n'_k + m_i + m'_k + 1) + T).$$

where T is the time required to find the sets of states satisfying p_{ij}, p'_{ij}, etc.

4.2. Determining ϕ_F and $\neg\phi_F$ for Büchi, Muller, Rabin, Streett, L and $\forall$ automata

Let $M = (S, s_0, \delta, F)$ be a Büchi, Muller, Rabin, Streett, L or $\forall$-automaton. With each of these six different types of ω–automata, we first give the acceptance condition and its negation as CTL* path formulas. Then we state the values of n, n_i, and m_i that show why the formulas have the general form above.

- Büchi

  $\phi_F = \mathbf{GF}(\bigvee_{s\in F} s)$

  $n = 1,\ n_1 = 0,\ m_1 = 1$

  $\neg\phi_F = \mathbf{FG}(\bigvee_{s\in F} s)$

  $n = 1,\ n_1 = 0,\ m_1 = 0$

- Muller

$$\phi_F = \bigvee_{A\in F} (\mathbf{FG}(\bigvee_{s\in A} s) \wedge \bigwedge_{s\in A} \mathbf{GF}s) \text{ or}$$

$n = \mid F \mid$ and for every $A \in F \;:\; n_A = 0,\; m_A = \mid A \mid$

Although it is possible to obtain a correct formula for $\neg\phi_F$ by simply negating ϕ_F and converting the result to the desired form, exponential growth can occur. Therefore, a formula is produced using Lemma 1:

$$\neg\phi_F = (\bigwedge_{A\in F} \mathbf{GF}(\bigvee_{s\in\overline{A}} s)) \vee \bigvee_{A\in F}\bigvee_{t\in A}(\bigwedge_{\substack{B\subset A\\ B\in F}} \mathbf{GF}(\bigvee_{s\in\overline{B}} s) \wedge \mathbf{FG}(\bigvee_{\substack{s\in A\\ s\neq t}} s))$$

$n = 1 + \Sigma_{A\in F} \mid A \mid,\; n_1 = 0,\; m_1 = \mid F \mid$

and for every $A \in F,\; t \in A \;:\; n_{A,t} = 0,\; m_{A,t} = \mid \{B \subset A \mid B \in F\} \mid$

- Rabin

$$\phi_F = \bigvee_{(U,V)\in F} (\mathbf{FG}(\bigvee_{s\in U} s) \wedge \mathbf{GF}(\bigvee_{s\in V} s))$$

$n = \mid F \mid,\; n_{(U,V)} = 0,\; m_{(U,V)} = 1$

$$\neg\phi_F = \bigwedge_{(U,V)\in F} (\mathbf{GF}(\bigvee_{s\in\overline{V}} s) \vee \mathbf{FG}(\bigvee_{s\in\overline{U}} s))$$

$n = 1,\; n_1 = \mid F \mid,\; m_1 = 0$

- Streett

$$\phi_F = \bigwedge_{(U,V)\in F} (\mathbf{FG}(\bigvee_{s\in U} s) \vee \mathbf{GF}(\bigvee_{s\in V} s))$$

$n = 1,\; n_1 = \mid F \mid,\; m_1 = 0$

$$\neg\phi_F = \bigvee_{(U,V)\in F} (\mathbf{GF}(\bigvee_{s\in\overline{U}} s) \wedge \mathbf{FG}(\bigvee_{s\in\overline{V}} s))$$

$n = \mid F \mid,\; n_{(U,V)} = 0,\; m_{(U,V)} = 1$

- L

$$\phi_F = \bigvee_{U\in Z} \mathbf{FG}(\bigvee_{s\in U} s) \vee \mathbf{GF}(\bigvee_{s\in V} s)$$

$n = 1 + \mid Z \mid,\; n_1 = 0,\; m_1 = 1$ and for all $U \in Z \;:\; n_U = 0,\; m_U = 0$

$$\neg\phi = \mathbf{FG}(\bigvee_{s\in\overline{V}} s) \wedge \bigwedge_{U\in Z} \mathbf{GF}(\bigvee_{s\in\overline{U}} s)$$

$n = 1,\; n_1 = 0,\; m_1 = \mid Z \mid$

- $\forall$

$$\phi_F = \mathbf{FG}(\bigvee_{s\in U} s) \vee \mathbf{GF}(\bigvee_{s\in V} s)$$

$$n = 1,\ n_1 = 1,\ m_1 = 0$$

$$\neg\phi_F = \mathbf{GF}(\bigvee_{s\in\overline{U}} s) \wedge \mathbf{FG}(\bigvee_{s\in\overline{V}} s)$$

$$n = 2,\ n_1 = 0,\ m_1 = 1,\ n_2 = 0,\ m_2 = 0$$

4.3. The complexity of the containment problem

The complexity of the containment problem for all possible combinations of ω-automata considered can now be obtained from the general result stated in 4.1. It is easy to see that the time required to label the states with the propositional subformulas p_{ij}, q_{ij}, etc. is dominated by the other terms of the complexity formula. To obtain each entry in the matrix we substitute the values of n, n_i, and m_i for the acceptance condition of the first automaton and n', n'_k, and m'_k for the negation of the acceptance condition for the second automaton.

5. Directions for Future Research

An obvious question is whether there are any reasonable acceptance conditions for ω–automata that we have not considered. Certainly we have included all of the models that are commonly discussed in the literature, but are there any that don't fit in our framework? One possibility is to use a formula of linear-temporal logic as the acceptance condition for an automaton. For example, given an arbitrary formula f of linear-temporal logic one can define an f-automaton that accepts an infinite word iff the word satisfies the formula f. The same question can be posed for Wolper's logic *ETL* [18] and, in fact, for any temporal logic with models that are sequences of states. This notion of acceptance at first appears more general than the previous ones that we have discussed, but it is really not. Assume that the alphabet for f is Σ. By using a construction of [17] it is possible to obtain a nondeterministic Büchi automaton with alphabet $\mathcal{P}(\Sigma)$ that will accept an infinite word iff the word satisfies the formula f. Consequently, this problem is essentially the same as the problem of showing containment between some (possibly nondeterministic) ω–automaton and a nondeterministic Büchi automaton.

The question of how to handle the containment problem when both automata are nondeterministic, is another important problem for research. In this case, the problem is at least PSPACE hard, since the containment problem for conventional automata on finite words has this complexity. In some cases it is possible to show that the containment problem for certain types of ω–automata is also in PSPACE. We conjecture that this is true for all of the cases in the complexity matrix, but we have not been able to prove this result yet. Of course, to say that a particular problem is in PSPACE is not really very useful in practice. Concrete time bounds like 2^n or $2^{n \log n}$ are much more useful. We have already obtained some results of this type and we hope to complete the matrix for the nondeterministic case with bounds of this sort in the near future.

M \ M'	Büchi det	Muller det	Rabin det	Streett det	L det	$\forall$ nondet
Büchi nondet	$ee' + vv'$	$ee'g' + vv'f'g'$	$ee'f' + vv'f'^2$	$ee'f' + vv'f'$	$ee' + vv'f'$	$ee' + vv'$
Muller nondet	$ee' + vv'g$	$ee'fg'$ $+vv'(ff'g' + gg')$	$ee'ff'$ $+vv'(ff'^2 + gf')$	$ee'ff' + vv'gf'$	$ee' + vv'(ff' + g)$	$ee'f + vv'g$
Rabin nondet	$ee'f + vv'f$	$ee'fg' + vv'ff'g'$	$ee'ff' + vv'ff'^2$	$ee'ff' + vv'ff'^2$	$ee'f + vv'ff'$	$ee'f + vv'f$
Streett nondet	$ee'f + vv'f^2$	$ee'fg'$ $+vv'f(f + f')g'$	$ee'(f + f')$ $+vv'(f + f')^2$	$ee'ff' + vv'f^2f'$	$ee'f + vv'f(f + f')$	$ee' + vv'f$
L nondet	$ee'f + vv'f$	$ee'fg' + vv'ff'g'$	$ee'ff' + vv'ff'^2$	$ee'ff' + vv'ff'$	$ee'f + vv'ff'$	$ee'f + vv'f$
$\forall$ det	$ee' + vv'$	$ee'f' + vv'g'$	$ee'f' + vv'f'$	$ee'f' + vv'f'$	$ee' + vv'f'$	$ee' + vv'$

Time complexity of the containment $\mathcal{L}(M) \subseteq \mathcal{L}(M')$

where $e = |\delta|$, $e' = |\delta'|$, $v = |S|$, $v' = |S'|$, $f = |F|$, $f' = |F'|$, and $g = \sum_{A \in F} |A|$, $g' = \sum_{A \in F'} |A|$ if the automaton is a Muller automaton.

Finally, although our algorithms for testing containment are the best that we know, we are currently unable to show that many of them are optimal. In fact, we suspect that some are not optimal and may be improved in the future. We believe that additonal research would be valuable in this direction. To aid in the search for better algorithms, it would be quite helpful to have lower bounds for the non-optimal cases in the matrix. A related question that we have not fully considered is the usefulness of our present complexity measure. Currently, the entries in our matrix are worst case execution times. While this measure is certainly an important factor in evaluating the efficiency of our algorithms, it is not the only factor of interest. Since the automata may have many thousands of states, the amount of memory available is frequently the limiting factor rather than the execution time. Complexity measures that take this into account, are probably more useful than time complexity alone. Devising such measures and finding algorithms that are efficient with respect to the new measures are also important directions for research.

References

[1] A. Arnold and P. Crubille. *A Linear Algorithm to Solve Fixed-Point Equations on Graphs.* Technical Report I-8632, Universite de Bordeaux, November 1986.

[2] E. M. Clarke and E. A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In Dexter Kozen, editor, *Proc. Workshop on Logic of Programs*, pages 52–71, Springer-Verlag: LNCS 131, Yorktown Heights, NY., 1981.

[3] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[4] D. L. Dill and E. M. Clarke. Automatic verification of asynchronous circuits using temporal logic. *IEE Proceedings*, 133, part E(5), Sep 1986.

[5] E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *JCSS*, 30(1):1–24, 1985.

[6] E. A. Emerson and C. L. Lei. Temporal reasoning under generalized fairness constraints. In *Springer LNCS 210*, STACS86, Orsay, France, January 1986.

[7] J. R. Büchi. On a decision method in resticted secon-order arithmetics. In *Proceedings, International Congres on Logic Method and Philosophy of Science, 1960*, pages 1–12, Stanford University Press, 1962.

[8] R. P. Kurshan. Complementing Deterministic Büchi Automata in Polynomial Time. *JCSS*, 35:59–71, 1987.

[9] R. P. Kurshan. *Testing Containment of ω-Regular Languages.* Technical Report 1121-861010-33-TM, Bell Laboratories, 1986.

[10] Z. Manna and A.Pnueli. Specification and verification of concurrent programs by $\forall$-automata. In *Proceedings - Fourteenth Annual ACM Symposium on Principles of Programming Languages, 1987*, pages 1–12, ACM, 1987.

[11] D. E. Muller. Infinite sequences and finite machines. In *Switching Cicuit Theory and Logical Design: Proceedings, Fourth Annual Symposium*, pages 3–16, 1963.

[12] M. Nivat. *Behaviours of Synchronized Systems of Processes.* Technical Report 81-64, Universite Paris 7, November 1981.

[13] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions, American Mathematical Society*, 141:1–35, 1969.

[14] S. Safra. On the complexity of ω–automata. In *Symposium on Foundations of Computer Science*, IEEE, Oct 1988.

[15] A. P. Sistla and E. M. Clarke. Complexity of propositional temporal logics. *Journal of the Association of Computing Machinery*, 32(2):733–749, 1986.

[16] R. S. Streett. Propositional dynamic logic of looping and converse is elementary decidable. *Information and Control*, 54:121–141, 1982.

[17] M. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proceedings of the Conference on Logic in Computer Science*, Boston, Mass., June 1986.

[18] P. Wolper. Temporal logic can be more expressive. *Inf. Control*, 56:72–79, 1983.