Information Processing Letters 46 (1993) 301-308
Elsevier

26 July 1993

A unified approach for showing language inclusion and equivalence between various types of ω-automata *

E.M. Clarke and I.A. Draghicescu
Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

R.P. Kurshan
AT&T Bell Laboratories, Murray Hill, NJ 07974, USA

Communicated by L. Kott
Received 23 July 1991
Revised 9 September 1992

Abstract

Clarke, E.M., I.A. Draghicescu and R.P. Kurshan, A unified approach for showing language inclusion and equivalence between various types of ω-automata, Information Processing Letters 46 (1993) 301-308.

We consider the language inclusion and equivalence problems for six different types of ω-automata; Büchi, Muller, Rabin, Streett, the L-automata of Kurshan, and the $\forall$-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 ith row and jth column is the complexity of showing inclusion between the ith type of automaton and the jth. Thus, for example, we give the complexity of showing language inclusion 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.

Keywords: Formal languages; ω-automata; language inclusion; temporal logic; computation tree logic; model checking

1. Introduction

ω-Automata were first used by Büchi in a paper on the decision problem for the logic S1S [2]. A short time later Muller showed that such automata were also useful for modeling the behavior of asynchronous circuits [7]. 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

Correspondence to: E.M. Clarke, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA.

* This research was partially supported by NSF grant CCR-87-226-33.

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 a 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 [10], Streett [13], Kurshan [4], and Manna and Pnueli [6]. It can be shown that each type of automaton accepts the same class of languages (i.e. the ω-regular languages [1]); however, the translation from one type of automaton to another may be quite complex [11].

The language inclusion 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. $\mathscr{L}(M_i)$ will be the language accepted by M_i. The *language inclusion problem* (or simply the *inclusion* problem when this is unambiguous) is the problem of determining whether $\mathscr{L}(M_1) \subseteq \mathscr{L}(M_2)$. The *equivalence problem*, on the other hand, is the problem of deciding whether $\mathscr{L}(M_1) = \mathscr{L}(M_2)$. Given an algorithm for the inclusion problem, we can easily obtain an algorithm for the equivalence problem, since $\mathscr{L}(M_1) = \mathscr{L}(M_2)$ iff $\mathscr{L}(M_1) \subseteq \mathscr{L}(M_2)$ and $\mathscr{L}(M_2) \subseteq \mathscr{L}(M_1)$. If M_2 is nondeterministic, then determining whether $\mathscr{L}(M_1) \subseteq \mathscr{L}(M_2)$ 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.

We consider the problem of deciding inclusion 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 Fig. 1). The entry in the ith row and jth column is the complexity of showing inclusion between the ith type of automata and jth. The entries on the diagonal of the matrix give the complexity of deciding inclusion 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.

Although some of our results were previously known (see [5] for instance), most of our results are new, because no one else has considered the hybrid cases (Büchi included 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 inclusion 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.

2. ω-Automata

A (*nondeterministic*) *ω-automaton* over an *alphabet* Σ is a tuple $M = (S, s_0, \delta, F)$ where S is a finite set of *states*, s_0 is an *initial state*, $\delta : S \times \Sigma \to \mathscr{P}(S)$ is a *transition relation* and F is an *acceptance condition*. The automaton is *deterministic* if $\forall s \in S$, $\forall a \in \Sigma$: $|\delta(s, a)| \leqslant 1$. The automaton is *complete* if $\forall s \in S$, $\forall a \in \Sigma$: $|\delta(s, a)| \geqslant 1$. 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 inclusion.

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 $\forall i \geqslant 1$, $\exists a_i \in \Sigma$: $\delta(s_i, a_i) \ni s_{i+1}$. 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 $\forall i \geqslant 1$: $\delta(s_i, a_i) \ni s_{i+1}$.

[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.

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 ∀-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 $M = (S, s_0, \delta, F)$ 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) $\inf(r) \subset A \setminus \{t\}$.

Proof. The "$\Rightarrow$" direction is proved by the following argument: Suppose that $\inf(r) \notin F$. Then either $\inf(r)$ is not included in any set of F, and in this case (1) holds, or $\inf(r)$ is included in some set in F. Let A be a minimal set in F such that $\inf(r) \subset A$. Then (2a) holds and as $\inf(r)$ must be strictly included 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 $\inf(f)$ 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 $F = \{(U_1, V_2), \ldots, (U_n, V_n)\}$, where $U_i, V_i \subseteq S$. A run is accepted by M if there exists $i \in \{1, \ldots, n\}$ such that $\inf(r) \cap U_i$ 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 $\inf(r) | U$ for some $U \in Z$ or $\inf(r) \cap V \neq \emptyset$.

The acceptance condition of a ∀-automaton is $F = (U, V) \subset S \times S$. 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 $\neg f$, $f \vee f$, $\mathbf{X}f$, 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 $K = (\mathcal{S}, \mathcal{R}, \mathcal{L})$, 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 $s_1 \to s_2$ to indicate that $(s_1, s_2) \in \mathcal{R}$.
- $\mathcal{L} : \mathcal{S} \to \mathcal{P}(\text{AP})$ 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 K* to be a sequence of states, $\pi = s_0s_1\ldots$ such that for every $i \geqslant 0$, $s_i \to s_{i+1}$. 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: $K,s \models f$ means that f holds at state s in structure K. Similarly, if f is a path formula, $K,\pi \models f$ means that f holds along path π in structure K. 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 $k \geqslant 0$ such that $\pi^k \models g_2$ and for all $0 \leqslant j < k$, $\pi^j \models g_1$.

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

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

Let $K = (\mathscr{S}, \mathscr{R}, \mathscr{L})$ be a finite Kripke structure. The *model checking problem* for a logic L is the problem of determining which states in $\mathscr{S}$ satisfy a given formula f of L. This problem is PSPACE-complete for CTL* [12]. However, for restricted CTL* formulas of the form

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

where p_{ij} and q_{ij} are propositional formulas, Emerson and Lei [3] give a polynomial model checking algorithm.

Theorem 2. *Let $K = (\mathscr{S}, \mathscr{R}, \mathscr{L})$ 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*

$$\mathscr{O}\left(\sum_{i=1}^{n} n_i|\mathscr{R}| + \sum_{i=1}^{n} n_i^2|\mathscr{S}| + T\right),$$

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

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

In this case the complexity of the model checking problem is

$$\mathrm{O}\left(\left(\sum_{i=1}^{n}(n_i+1)+1\right)|\mathscr{R}| + \sum_{i=1}^{n}(n_i+1)(n_i+m_i+1)|\mathscr{S}| + T\right),$$

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 results 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

4.1. Reducing the inclusion 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 fullfills the acceptance condition

given by F if and only if it satisfies the formula 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

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

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

We now describe how to compute the complexity of the inclusion 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 ∀-automata over Σ such that $S\wedge S'=\emptyset$.

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

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

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

where ϕ_F expresses the acceptance 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 and only if and 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

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

$$\neg\phi_{F'}=\bigvee_{i=1}^{n'}\left(\bigwedge_{j=1}^{n_i'}(\mathbf{FG}p_{ij}'\vee\mathbf{GF}q_{ij}')\wedge\bigwedge_{j=1}^{m_i'}\mathbf{GF}r_{ij}'\wedge\mathbf{FG}p_i'\right).$$

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

$$K(M, M')\vDash\mathbf{E}\left[\bigvee_{i=1}^{n}\bigvee_{k=1}^{n'}\left(\bigwedge_{j=1}^{n_i}(\mathbf{FG}p_{ij}\vee\mathbf{GF}q_{ij})\wedge\bigwedge_{l=1}^{n_k'}(\mathbf{FG}p_{kl}'\vee\mathbf{GF}q_{kl}')\wedge\bigwedge_{j=1}^{m_i}\mathbf{GF}r_{ij}\wedge\bigwedge_{l=1}^{m_k'}\mathbf{GF}r_{kl}'\wedge\mathbf{Fg}(p_i\wedge p_k')\right)\right]$$

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

$$\mathrm{O}\left(\sum_{i=1}^{n}\sum_{k=1}^{n'}(n_i+n_k'+1)(|\delta|\cdot|\delta'|+|S|\cdot|S'|(n_i+n_k'+m_i+m_k'+1)+T)\right),$$

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 ∀-automata

Let $M=(S, s_0, \delta, F)$ be a Büchi, Muller, Rabin, Streett, L or ∀-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

$$\phi_F = \mathbf{GF}\Big(\bigvee_{s\in F} s\Big), \quad \text{where } n=1,\ n_1=0,\ m_1=1.$$

$$\neg\phi_F = \mathbf{FG}\Big(\bigvee_{s\in \bar{F}} s\Big), \quad \text{where } n=1,\ n_1=0,\ m_1=0.$$

- Muller

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

where $n=|F|$ and for every $A\in F$: $n_A=0$, $m_A=|A|$.

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:

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

where $n=1+\Sigma_{A\in F}|A|$, $n_1=0$, $m_1=|F|$ and for every $A\in F$, $t\in A$: $n_{A,t}=0$, $m_{A,t}=|\{B\subset A\mid B\in F\}|$.

- Rabin

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

where $n=|F|$, $n_{(U,V)}=0$, $m_{(U,V)}=1$.

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

where $n=1$, $n_1=|F|$, $m_1=0$.

- Streett

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

where $n=1$, $n_1=|F|$, $m_1=0$.

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

where $n=|F|$, $n_{(U,V)}=0$, $m_{(U,V)}=1$.

- L

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

where $n=1+|Z|$, $n_1=0$, $m_1=1$ and for all $U\in Z$: $n_U=0$, $m_U=0$.

$$\neg\phi_F = \mathbf{FG}\Big(\bigvee_{s\in\bar{V}} s\Big)\wedge\bigwedge_{U\in Z}\mathbf{GF}\Big(\bigvee_{s\in\bar{U}} s\Big),$$

where $n=1$, $n_1=0$, $m_1=|Z|$.

M' / M	Büchi det	Muller det	Rabin det	Streett det	L det	∀ 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	$ee'f+vv'f$	$ee'fg'+vv'ff'g'$	$ee'ff'+vv'ff'^2$	$ee'ff'+vv'ff'$	$ee'f+vv'ff'$	$ee'f+vv'f$
∀ det	$ee'+vv'$	$ee'f'+vv'g'$	$ee'f'+vv'f'$	$ee'f'+vv'f'$	$ee'+vv'f'$	$ee'+vv'$

Fig. 1. 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 $g=\Sigma_{A\in F}|A|$, $g'=\Sigma_{A\in F'}|A|$ if the automaton is a Muller automaton.

- $\forall$

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

where $n = 1$, $n_1 = 1$, $m_1 = 0$.

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

where $n = 1$, $n_1 = 0$, $m_1 = 1$.

4.3. The complexity of the inclusion problem

The complexity of the inclusion problem for all possible combinations of ω-automata considered can now be obtained from the general result stated in Section 4.1 (see Fig. 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. In [9], six other types of automata are mentioned which we have not had an opportunity to consider. But the general question remains, is it possible to have a reasonable acceptance condition which does not fit into 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* [15] 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 [14] it is possible to obtain a nondeterministic Büchi automaton and hence a (potentially larger) deterministic Muller automaton with alphabet $\mathscr{P}(\Sigma)$ 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 inclusion between some (possibly nondeterministic) ω-automaton and a deterministic Muller automaton.

The question of how to handle the inclusion problem when both automata are nondeterministic, is another important problem for research. In this case, the problem is at least PSPACE hard, since the inclusion problem for conventional automata on finite words has this complexity. In some cases it is possible to show that the inclusion 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.

References

[1] A. Arnold and P. Crubille, A linear algorithm to solve fixed-point equations on graphs, Tech. Rept. I-8632, Université de Bordeaux, 1986.

[2] J.R. Büchi, On a decision method in resticted secon-order arithmetics, in: *Proc. Internat. Congres on Logic Method and Philosophy of Science*, 1960 (Stanford University Press, 1962) 1–12.

[3] E.A. Emerson and C.L. Lei, Temporal reasoning under generalized fairness constraints, in: *Proc. STACS86*, Lecture Notes in Computer Science **210** (Springer, Berlin, 1986).

[4] R.P. Kurshan, Testing containment of ω-regular languages, Tech. Rept. 1121-861010-33-TM, Bell Laboratories, 1986.

[5] R.P. Kurshan, Complementing deterministic Büchi automata in polynomial time, *J. Comput. System Sci.* **35** (1987) 59–71.

[6] Z. Manna and A. Pnueli, Specification and verification of concurrent programs by α-automata, in: *Proc. 14th Ann. ACM Symp. on Principles of Programming Languages, 1987* (ACM, New York, 1987) 1–12.

[7] D.E. Muller, Infinite sequences and finite machines, in: *Proc. Fourth Ann Symp. on Switching Cicuit Theory and Logical Design* (1963) 3–16.

[8] M. Nivat, Behaviours of synchronized systems of processes, Tech. Rept. 81-64, Université Paris 7, November 1981.

[9] M. Nivat and A. Saoudi, Automata on infinite objects and their applications to logic and programming, *Inform. and Comput.* **83** (1) (1989).

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

[11] S. Safra, On the complexity of ω-automata, in: *Proc. IEEE Symp. on Foundations of Computer Science*, 1988.

[12] A.P. Sistla and E.M. Clarke, Complexity of propositional temporal logics, *J. ACM* **32** (2) (1986) 733–749.

[13] R.S. Streett, Propositional dynamic logic of looping and converse is elementary decidable, *Inform. and Control* **54** (1982) 121–141.

[14] M. Vardi and P. Wolper, An automata-theoretic approach to automatic program verification, in: *Proc. Conf. on Logic in Computer Science*, Boston, MA, June 1986.

[15] P. Wolper, Temporal logic can be more expressive, *Inform. and Control* **56** (1983) 72–79.