ELSEVIER

Theoretical Computer Science 141 (1995) 69–107

Theoretical Computer Science

Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra

David E. Muller, Paul E. Schupp*

Department of Mathematics, University of Illinois, Urbana, IL 61801, USA

Dedicated to the memory of Ahmed Saoudi

Received February 1994; revised July 1994
Communicated by M. Nivat

Abstract

We give a proof that alternating tree automata can be simulated by nondeterministic tree automata which yields new complexity results and a unified proof of the theorems of Rabin, McNaughton and Safra. We also give a simple axiomatic framework for uniformizing strategies.

1. Introduction

We have discovered a new method which proves that alternating tree automata can be simulated by nondeterministic automata. Our method allows us to deduce very strong results about complexity and yields not only another proof of Rabin's theorem [14] on the decidability of the monadic theory of the tree but also the theorems of McNaughton [8] and Safra [17], and several new results. For example, we prove a general complexity result for complementing languages accepted by tree automata whose acceptance condition is given by pairs. (The result of Safra [18] on determinizing ω-automata using complemented pairs acceptance seems to be "dual" to our approach.) Indeed, we give a unified and fairly simple proof of the most fundamental results in the theory of automata working on infinite inputs. A fundamental breakthrough in really understanding Rabin's theorem was made by Gurevich and Harrington [6] and the reader familiar with their work will see our indebtedness to their approach. While we draw from them the fundamental idea of the "later appearance record", and while we also talk about strategies, the considerations of our proof are

* Corresponding author. Email: Schupp@math.uiuc.edu.

SSDI 0304-3975(94)00214-2

very different. We believe that alternating automata really provide the "natural model" for automata working on infinite inputs, and this is our main claim. Of course, other approaches are possible and there has been much recent work in this area. We mention the articles of Muchnick [10], Emerson and Jutla [4], Klarlund [7], McNaughton [9], Yaknis and Yaknis [22,23], and Zeitman [24].

We now briefly review our concept of an alternating automaton [12]. The reader unfamiliar with alternating automata may want to first read the more complete description of alternating automata on infinite trees in Appendix C. In Rabin's model of a nondeterministic automaton

$$M = \langle Q, \Sigma, \delta, q_0, \mathscr{F} \rangle$$

working on the binary tree, the transition function is a function

$$\delta : \Sigma \times Q \to \mathscr{P}(Q \times Q),$$

while in an alternating automaton the transition function is a function

$$\delta : \Sigma \times Q \to \mathscr{L}(K \times Q),$$

where K is the set of directions in the tree ($K = \{0,1\}$ for a binary tree) and $\mathscr{L}(K \times Q)$ is the free distributive lattice generated by all the pairs (d,q) where d is a direction and q is a state. For example, we would write a nondeterministic function

$$\delta(a, q_0) = \{(q_0, q_1), (q_2, q_3)\}$$

using the lattice notation as

$$\delta(a, q_0) = (0, q_0) \wedge (1, q_1) \ \vee \ (0, q_2) \wedge (1, q_3)$$

(where $\wedge$ has precedence over $\vee$ as usual). Intuitively, $\vee$ represents choice and $\wedge$ says "do both things". Thus there is a natural notion of a "deterministic" alternating automaton – namely an alternating automaton with no $\vee$'s in its transition function. A lattice expression can be written uniquely in *irredundant disjunctive normal form*. We show in Appendix C that the automaton may be regarded as acting on its input using any form for the function δ, and that the result does not depend upon the form in which the lattice element is written.

The main difference between nondeterministic and alternating automata is that if we dualize the transition function above by interchanging $\wedge$ and $\vee$ as usual and rewrite it again in irredundant disjunctive normal form we have

$$\tilde{\delta}(a, q_0) = (0, q_0) \wedge (0, q_2) \ \vee \ (0, q_0) \wedge (1, q_3) \ \vee \ (1, q_1) \wedge (1, q_3) \ \vee \ (0, q_2) \wedge (1, q_1),$$

which is not the transition function of a nondeterministic automaton, but is perfectly allowable as an alternating transition function.

Alternating automata are a sort of completion of nondeterministic automata. It is only by going to $\mathscr{L}(K \times Q)$ that one can always dualize transition functions. If

$$M = \langle Q, \Sigma, \delta, q_0, \mathscr{F} \rangle$$

is an alternating automaton then its *dual* is the automaton

$$\tilde{M} = \langle Q, \Sigma, \tilde{\delta}, q_0, \bar{\mathscr{F}} \rangle$$

obtained by dualizing the transition function and complementing the acceptance condition $\mathscr{F}$. If $\mathscr{F}$ is defined by subset acceptance then $\bar{\mathscr{F}}$ is also, but complementation of acceptance converts Büchi into co-Büchi and pairs into complemented pairs. The following basic fact about alternating automata was proved in [12] and is reproved more simply in Appendix C.

Theorem 1.1 (The Complementation Theorem). *The dual automaton $\tilde{M}$ accepts the complement of the language accepted by M.*

Since complexity depends on the acceptance condition used, we discuss acceptance conditions. The definition that a tree automaton M with state set Q accepts an input t is always that there exists a run ρ of M on t such that every individual history h of a copy of M which exists in ρ satisfies the given condition. In *Büchi acceptance* [1] one is given a subset $G \subseteq Q$ and h accepts if h contains states from G infinitely often. In *co-Büchi acceptance* one is given a subset $R \subseteq Q$ and h accepts if h contains states from R only finitely often. The *pairs condition* was introduced by McNaughton [8] and used by Rabin [14]. One is given a finite index set I and a collection $\Omega = \{(G_i, R_i)\}_{i \in I}$ of distinct pairs of disjoint subsets (G_i, R_i) of Q. The history h accepts if, for *some* $i \in I$, h contains states from G_i infinitely often *and* states from R_i only finitely often. The *complemented pairs*, or co-pairs condition, was first used by Streett [20]. One is again given a collection $\{(G_i, R_i)\}_{i \in I}$ of pairs, but now h accepts if for *every* $i \in I$, h either contains states from G_i infinitely often *or* states from R_i only finitely often. *Subset acceptance* was introduced by Muller [11]. Now, one is given a family $\mathscr{F} \subseteq \mathscr{P}(Q)$ and h accepts if $\inf(h)$, the set of states occurring infinitely often in h, is a member of $\mathscr{F}$. The complement of this condition is simply the subset condition defined by the complementary family $\bar{\mathscr{F}}$. The three conditions – pairs, complemented pairs and subset acceptance – are all equivalent in the sense that a tree automaton using any condition may be converted into another automaton using one of the others, but conversion to any except subset acceptance costs states. The automata defined by any one of these conditions accept exactly the family of recognizable languages.

Note that both Büchi and co-Büchi acceptance are special cases of an acceptance condition defined by a single pair or by a single complemented pair. The Büchi condition defined by the subset G is also defined by the single pair $(G, \emptyset)$ as well as the single complemented pair $(G, \bar{G})$. (If a history contains states from $\bar{G}$ only finitely often it certainly contains states from G infinitely often.) The co-Büchi condition defined by R is also defined by the single complemented pair $(\emptyset, R)$ and by the single pair $(\bar{R}, R)$. In general, the complement of the direct pair condition $\{(G_i, R_i)\}$ is the complemented pair condition $\{(R_i, G_i)\}$. Families of languages defined by Büchi and co-Büchi acceptance form proper subfamilies of those defined by single pairs and by single

complemented pairs which in turn form proper subfamilies of the recognizable languages.

If M is any automaton, we write $|M|$ for the number of states of M. We can now state our principal theorem.

Theorem 1.2 (The Simulation Theorem). *There is an effective construction which, when given an alternating automaton M on infinite trees produces an equivalent nondeterministic automaton N. If M uses acceptance defined by m complemented pairs, then $|N| \leqslant 2^{d|M|\log|M|}$, where d depends only on m, but is exponential in m. If the sets G_i in the complemented pairs form a chain then $|N|$ is no greater than $2^{bm|M|\log|M|}$ where b is a small fixed constant, and m is the number of complemented pairs. If M uses subset acceptance then $|N|$ is two exponentials in $|M|$. In any case, if M is deterministic, then N is also deterministic. The automaton N itself uses complemented pair acceptance. The number of complemented pairs is logarithmic in $|N|$. Finally, if M uses Büchi acceptance then N also uses Büchi acceptance.*

We shall see that the following is indeed a corollary.

Corollary 1.1 (Complexity of Complementation). *Let A be a nondeterministic (or indeed alternating) tree automaton. If A accepts with m pairs then there is a nondeterministic automaton N accepting the complementary language $L(A)$ with $|N| \leqslant 2^{d|A|\log|A|}$. If the sets R_i in the pairs form a chain then $|N| \leqslant 2^{bm|A|\log|A|}$. If A uses an arbitrary subset condition then $|N|$ is two exponentials in $|A|$. In any case, if A is a nondeterministic automaton on the line then N is deterministic.*

Alternating automata give a powerful uniform method for thinking about constructions in automaton theory. We illustrate this by showing how several basic theorems follow from the Simulation Theorem.

The correspondence between automata and formulas of monadic logic was first established by Büchi [1] in the case of the line (the natural numbers $\mathbb{N}$ with successor function). Büchi worked with nondeterministic automata using Büchi acceptance and proved closure under complementation. One cannot determinize using only Büchi acceptance. Muller [11] stated the principle that one could determinize if one used subset acceptance. McNaughton [8] then provided a complete proof that given any nondeterministic automaton N on the line using subset acceptance there is an equivalent deterministic automaton D using subset acceptance, and this result is called McNaughton's Theorem.

We have seen that a "deterministic" alternating automaton is defined as an alternating automaton with only $\wedge$ in its transition function. The Simulation Theorem says that any deterministic alternating automaton can be simulated by a deterministic automaton. Let us see why this yields McNaughton's Theorem. Let B be a nondeterministic automaton working on the line. If we write B as an alternating automaton

only $\vee$ occurs in its transition function. Thus only $\wedge$ occurs in the transition function of its dual $\tilde{B}$. The Simulation Theorem gives a deterministic automaton D' on the line equivalent to $\tilde{B}$. So D' accepts $\overline{L(B)}$. Since D' is on the line and is deterministic, we obtain an automaton D equivalent to B simply by complementing the acceptance condition of D'.

Much more recently, Safra [17] proved that if one starts with a Büchi automaton B, then the size of the state set of the equivalent deterministic D need only be one exponential in $|B|$. Keeping the notation of the paragraph above, if B is Büchi then $\tilde{B}$ is co-Büchi and the simulation theorem give D with $|D| \leqslant 2^{b|B|\log|B|}$, yielding Safra's Theorem.

Rabin [17] extended the correspondence between automata and monadic logic to the case of trees (the monadic theory S2S of two successor functions). This establishes the decidability of S2S – one of the most important positive solutions of a decision problem. The most difficult step was establishing closure under complementation. If A is a tree automaton then $\tilde{A}$ is an alternating automaton which accepts $\overline{L(A)}$. The Simulation Theorem then gives a nondeterministic tree automaton N equivalent to $\tilde{A}$, which thus accepts $\overline{L(A)}$. But it also gives the complexity result that if A uses pairs acceptance then $|N| \leqslant 2^{d|A|\log|A|}$ where d depends only on the number of pairs. This supports the notion that the number of pairs is a reasonable complexity measure.

To establish the decidability of S2S it is also necessary to show that there is an algorithm which, when given a nondeterministic automaton A, decides whether or not the language $L(A)$ is empty. If A is a nondeterministic automaton working on k-ary trees labeled from an alphabet Σ, first perform the standard trick of projecting to the one-letter alphabet Σ_1 and let A_1 be the corresponding automaton. Now, a nondeterinistic automaton working on k-ary trees labeled from Σ_1 *is* the same thing as an *alternating* automaton working on the line labeled by Σ_1 – it simply sends out lots of copies. Considering A_1 as an alternating automaton on the line, the Simulation Theorem yields an equivalent nondeterministic automaton N. But deciding emptiness for a nondeterministic automaton on the line is trivial.

Rabin [15] also proved a remarkable characterization of languages definable by a formula of weak monadic logic (where only quantifiers over finite sets are allowed): A language L is weakly definable if and only if both L and its complement $\bar{L}$ are accepted by Büchi automata. We discussed how alternating automata clarify this result in our paper [13] with Ahmed Saoudi. Rabin [15] gives a short conceptual proof that if L and $\bar{L}$ are both accepted by Büchi automata then L is weakly definable. The other direction is much longer because it is hard to calculate with Büchi automata.

In [13], we define a "weak" acceptance condition which can be thought of as a very special case of Büchi acceptance and is such that the complement of the acceptance condition is a condition of the same type. Michel Parigot pointed out to us that we actually proved the general principle that an alternating automaton using Büchi acceptance can be simulated by a nondeterministic automaton using Büchi acceptance. It is easy to prove that any weakly definable language L is accepted by a weak

alternating automaton. Assuming this, then L and $\bar{L}$ are accepted by weak alternating automata M and $\tilde{M}$ by the Complementation Theorem. Thus both L and $\bar{L}$ are accepted by Büchi automata by the Simulation Theorem. So L is weakly definable by the short part of Rabin's proof.

Another result which we consider basic to the general theory of automata on infinite inputs is Rabin's Regularity Theorem [16]: A nonempty regular language of k-ary trees contains a regular tree. Of course, closure under complementation, the decidability of the emptiness problem and the regularity theorem also follow from Gurevich and Harrington. Indeed, their determinacy result says that if, as above, we project to the one-letter alphabet and consider the automaton A_1, then there is an accepting run of A_1 in which two copies which have the same LAR make the same transition. Pulling this run back to the original alphabet defines a regular tree in the language. Appendix D gives a proof only slightly more complicated using our results.

We turn to a more detailed discussion of the relationship of our method to the Gurevich–Harrington Theorem. Acceptance of an input tree t by an automaton M can be expressed as the existence of a winning first-player strategy in a certain infinite game $\Gamma(M,t)$ – the "acceptance game" of M on t. See [5] and Section 2 for details and for a precise definition of a strategy, but the game for an alternating automaton is very similar to the game for a nondeterministic automaton. Büchi [2] had already formulated the principle that the strategy of a game corresponding to an automaton should not need to be "too complicated". Gurevich and Harrington [6] provided a precise measure, the later appearance record (LAR) of the bounded quantity of memory necessary in the case of an automaton using subset acceptance. They proved the Forgetful Determinacy Theorem: If a nondeterministic automaton accepts, then it has a winning strategy S in which two copies of the automaton which have the same LAR and the same future (isomorphic subtrees in front of them) can make the same transition.

Our result is weaker than that of Gurevich and Harrington for subset acceptance in the following way. From the standpoint of alternating automata we are not required to worry about when subtrees are isomorphic. An alternating automaton already organizes together those copies which we need to consider – namely the ones at the same vertex. This simplifies the proof.

It is remarkable that the LAR is always a sufficient quantity of memory for any acceptance condition. We want to investigate the question of how much memory is needed in the case of certain special acceptance conditions. We develop a simple axiomatic framework for addressing the question of how much memory a strategy needs. Our method basically works with complemented pairs acceptance. We show how taking LAR's converts subset acceptance into a different automaton using complemented pairs acceptance. If one is given an automaton whose acceptance condition is already defined by complemented pairs, then one needs less memory – at least in the case where the number of co-pairs is less than the size of the state set. For this situation we show that an appropriate memory is the index appearance record, which is a sort of LAR but which uses the index set of the co-pairs instead of the state

set. This difference is significant for our results on complexity. An important special case is when the collection $\{G_i\}_{i\in I}$ of "green sets" occurring in the co-pairs forms a chain. In this situation no memory beyond the current state is necessary. Considering the above cases allows us to formulate a general axiomatic definition of when a memory is sufficient.

We are interested in uniformizing strategies in a relatively effective way. If $\Gamma(M,t)$ is the acceptance game of the alternating automaton M on the tree t and we have an appropriate memory for M and the first player has a winning strategy, then it has a winning strategy which depends only on the appropriate memory and the position on the input tree. Uniformizing over fewer copies also allows us to prove that there is a uniform strategy $\hat{S}$ which is effectively calculable in a very strong sense *relative* to any given winning strategy S. We emphasize that S itself may be highly nonrecursive and that we are talking about relative calculability with an oracle for the values of S.

Let S be a first-player strategy for the acceptance game $\Gamma(M,t)$ of an alternating automaton M on an input tree t. We say that S is *zero memory* if, for all finite histories h, $S(h)$ depends only on the position $v(h)$ of h and the last state occurring in h. S is *μ-uniform* if $S(h)$ depends only on $v(h)$ and a bounded quantity $\mu(h)$ of "memory" which can be calculated by a finite automaton reading the history h. A strategy $\hat{S}$ is a *uniformization* of a strategy S if, for all h, $\hat{S}(h)$ is the same as $S(h')$ for some history h' with the same vertex and the same memory as h. By saying that $\hat{S}$ is *almost finite state* in S we mean that we calculate $\hat{S}$ from S by the following simple finite recursion scheme. There is a fixed bound C so that at any stage we keep a list of not more than C histories generated by S. In response to a move by the second player we must consult an oracle for values of S on the histories in our list. Given these values, the value of $\hat{S}$ and which histories to carry on to the next stage are decided by a finite automaton using only the values of S on the memories of the histories in the list. Such a relative effectiveness result is not true in the Gurevich–Harrington situation since the question of which subtrees are isomorphic may be more complicated than the original strategy S of the automaton. Our method of proof yields the following result about strategies.

Theorem 1.3 (The Strategy Uniformization Theorem). *Let S be any winning first-player strategy in the acceptance game $\Gamma(M,t)$ where M is an alternating automaton and let μ be the appropriate memory. Then the first player has a μ-uniform winning strategy $\hat{S}$ which is a uniformization of S and which is almost finite state in S in the sense described above.*

The uniform strategy need not be any more complicated than any given winning strategy. (See also the illuminating remarks of Gurevich [5].) We always work from the point of view of the first player OR and the above result applies to all the cases which we study.

Now Martin's Theorem ensures that every acceptance game $\Gamma(M,t)$ is *determined*, that is, one of the two players has a winning strategy. Thus if OR does not have

a winning strategy then AND must have one. In Appendix C we prove the Duality Principle which says that a winning strategy for AND for $\Gamma(M,t)$ directly yields a winning first-player strategy in the dual game $\Gamma(\tilde{M},t)$. But dualizing the game complements the acceptance condition and thus the same memory may no longer be appropriate. A "forgetful determinacy" statement asserts that one of the two players has a winning strategy which depends only on the given memory and the input tree. We prove this in two cases. First, as discovered by Gurevich and Harrington, this holds in the case of subset acceptance where the memory is the LAR. This case is perfectly general in the sense that *any* acceptance condition can be represented as such a subset acceptance condition. Second, we obtain the result that if the acceptance condition is defined by a set of complemented or direct pairs $\{(G_i,R_i)\}$, $i=1,\ldots,n$, satisfying the "double chain" condition $G_1\subset\cdots\subset G_n$ and $R_1\subset\cdots\subset R_n$ then one of the players has a winning strategy which depends only on position and the last state.

We have tried to make this paper as clear as possible. In Section 2 we discuss how the amount of memory which a strategy needs depends on its acceptance condition and develop the axiomatic framework. In Section 3 we discuss our method of uniformizing strategies. This constitutes the real novelty of our approach. Intuitively, once one knows that an alternating automaton M can always accept with a bounded memory strategy, one *should* be able to simulate it with a nondeterministic automaton N. We carry out the construction of N in Sections 3 and 4, paying attention to its size. Appendix I shows that an automaton whose acceptance condition is defined by two complemented pairs need not have any zero-memory winning strategy. Appendix B shows how alternating automata can be viewed as nondeterministic automata running on covering trees. Appendix C reviews some essential notation and results concerning alternating automata on infinite trees, including the Logical Equivalence Theorem showing that the properties of an alternating automaton are not affected by the form of the logical expression used for defining its transition function, which is used to establish the Duality Principle mentioned earlier. Appendix D proves Rabin's Regularity Theorem. Finally, although our interest in developing a general definition of a memory framework is to investigate when one can use *less* memory, Appendix E shows that our strategy uniformization result applies to infinite automata.

2. Subset acceptance, appearance records and complemented pairs – How much memory does a strategy need?

Given an alternating automaton M on an input tree t, we consider the *acceptance game*, $\Gamma(M,t)$ of M on t. The game has players OR, sometimes called the first player, and AND, sometimes called the second player (where, intuitively, OR plays for acceptance and AND plays for rejection). Let us assume that the expression for $\delta(a_0,q_0)$ is written in irredundant disjunctive normal form. In the first move, OR chooses a term τ_1 from $\delta(a_0,q_0)$ where a_0 is the letter labeling the origin of t. Then AND chooses a generator (k_1,s_1) occurring in the term chosen by OR. At stage $n+1$,

the sequence $h_n = q_0\tau_1 k_1 s_1 \cdots \tau_n k_n s_n$ already chosen by the players is called an *n-history* or *initial play*.[1] Let v_n be the *position* or *vertex* on the input tree t at the end of a finite path $\lambda_n = k_1 \ldots k_n$ which is associated with the history h_n. Let a_n be the letter labeling the vertex v_n of t. Now OR chooses a term τ_{n+1} from $\delta(a_n, s_n)$ and AND subsequently chooses a generator (k_{n+1}, s_{n+1}) in the term chosen by OR. If either player violates the rules restricting its choices it immediately loses. Assuming that both players follow the rules, the infinite sequence of choices defines an *infinite history*, or *infinite play*, $h = q_0\tau_1 k_1 s_1 \ldots$ Player OR wins the play if the *state sequence* $q_0 s_1 \ldots s_n \ldots$ satisfies the acceptance condition of M. Otherwise, AND wins.

A *strategy* S for OR, consists of a rule, not necessarily recursive, which determines a choice for OR at each step in the game.[2] This choice must depend only upon the initial play h_n up to the point where the choice is to be made, and upon the tree t, but *not* upon any subsequent choices which are made by either player. A strategy S for OR is called a *winning strategy* if all possible plays which can be generated by following the strategy S are won by OR. A similar definition is made for a winning strategy for AND. Now, M *accepts* the input t if and only if OR has a winning strategy for $\Gamma(M, t)$. The reader may either take this as a definition or read the proof of the equivalence of the existence of a winning strategy and the computation tree definition of acceptance [12].

From the point of view of automaton theory, the crucial question is how much information from the past a strategy needs to use in order to be a winning strategy whenever such a strategy exists. One of the important accomplishments of the work of Gurevich and Harrington was to provide a specific quantity of memory – the "later appearance record" or LAR – which is always sufficient.

We first discuss LAR's and show how taking LAR's converts subset acceptance into complemented pairs acceptance. This gives a very simple proof that an automaton using subset acceptance is equivalent to one using complemented pairs acceptance. The reader familiar with the work of Gurevich and Harrington will note that there is a slight difference in our definition of an LAR. We record slightly more information, but this is important to our proof.

We use LAR in two senses. First, it represents a function from the set of strings on some set Q to the set of sequences without repetitions on the set $Q \cup \{\emptyset\}$. Second, we use "LAR" to represent a string which can be the image under this function and hence any sequence from the set $Q \cup \{\emptyset\}$ which is without repetitions and does not end in $\emptyset$. The LAR function is given by the following definition.

---

[1] One can consider games where the expression for δ may not be written in irredundant disjunctive normal form, but is in a general parenthesized formula which may be thought of as a rooted tree whose vertices have labels $\vee$ and $\wedge$. Then OR chooses the disjuncts ($\vee$-connected subformulas) and AND chooses the conjuncts ($\wedge$-connected subformulas) until a generator (k, s) is reached. It is consequently shown that the terms τ_i may be omitted from the play without affecting the results. See Appendix C.

[2] Formally, S is a function from the set $\mathscr{H}$ of all possible finite histories to the set $\mathscr{T}$ of terms of the transition function of M.

Definition 2.1. The LAR of the single letter q is q. If $\mathrm{LAR}(\gamma) = \zeta$ and γ is extended to the string γq then the LAR of γq is formed as follows. First, remove the symbol $\emptyset$ from ζ if it occurs, to obtain the string ζ'. Second, add q to the right-hand end of ζ'. If q occurs in ζ' then replace that occurrence by the symbol $\emptyset$. (For example, if $\mathrm{LAR}(\gamma) = \emptyset q_1 q_3 q_2$ then $\mathrm{LAR}(\gamma q_3) = q_1 \emptyset q_2 q_3$.)

When constructing the LAR function the result will contain $\emptyset$ unless the last letter q is a new letter which has not occurred before.[3] We denote the set of all possible LAR's by $\mathscr{L}$.

Definition 2.2. Let $\zeta \in \mathscr{L}$ be an LAR and let F be a subset of Q. We say that ζ *displays* F if there is a suffix ζ' of ζ such that F consists exactly of those letters, excluding $\emptyset$, occurring in ζ'. Furthermore, we say ζ *shifts* F if it displays F and the suffix ζ' consists of exactly those letters to the right of the symbol $\emptyset$ in ζ.

Thus, for example, if $F = \{q_1, q_2\}$ then $q_0 q_1 \emptyset q_2$ displays F but does not shift F, while $q_0 \emptyset q_2 q_1$ shifts F.

The LAR depends only on a sequence of states. But, in general, histories are really the sequences which have memories. If $h_n = q_0 \tau_1 k_1 s_1 \ldots \tau_n k_n s_n$ is an n-history we set $\zeta_n = \mathrm{LAR}(q_0 s_1 \ldots s_n)$. We say that $\{\zeta_n\}_{n=1}^{\infty}$ is the sequence of LAR's *associated* with h. The connection between the previous definition and subset acceptance is given by the following lemma.

Lemma 2.1. *Let h be an infinite history and let $\{\zeta_n\}_{n=1}^{\infty}$ be the associated sequence of LAR's. Then* $\inf(h) = F$ *if and only if F is always displayed by ζ_n from some subscript n onwards and F is shifted infinitely often.*

Proof. First, let $F = \inf(h)$. Then only states of F occur in h from some point onwards and all states of F occur infinitely often. Choose an index n_0 such that only states of F occur after n_0. Choose n_1 such that all states of F have been repeated after n_0. Then F is displayed in all ζ_n with $n > n_1$. To check that F is shifted infinitely often, choose $n > n_1$ and write $\zeta_n = \zeta'\zeta''$ where ζ'' is the suffix of ζ_n displaying F. If q is the first state occurring in ζ'' then F will be shifted at the next repeat of q. Repeating the argument after this shift, we see that F will be shifted infinitely often.

Now suppose that F is displayed from some point onwards and that F is shifted infinitely often. Choose n_0 such that ζ_n displays F for all $n > n_0$. Certainly, no state q *not* in F can then occur, since an LAR ζ_n always ends in the last state to occur and

[3]The LAR function is "reverse sequential" in the sense that it can be printed by a finite automaton with output, which reads the history in reverse (from final state to initial) and has the LAR as its output, also printed in reverse. This automaton prints a state it reads only if it is a new state which has not been encountered before in the sequence, and it prints $\emptyset$ if it encounters the final state a second time.

a suffix ending in q cannot display F. Suppose that some $q \in F$ occurred only finitely often. Take n_2 such that there has been a repetition of states after the last occurrence of q. Then q is always strictly to the left of $\emptyset$ in all ζ_n with $n > n_2$. Thus, F could not have been shifted by any ζ_n with $n > n_2$. □

Definition 2.3. Let $M = \langle Q, \Sigma, \delta, q_0, \mathscr{F} \rangle$ be an automaton with final family $\mathscr{F}$. Write the complement $\bar{\mathscr{F}}$ as $\bar{\mathscr{F}} = \{F_1, \ldots, F_m\}$, and let $I = \{1, \ldots, m\}$. Let G_i consist of those LAR's which do not display F_i, let Y_i consist of those LAR's which display but which do not shift F_i and let R_i consist of those LAR's which shift F_i.

If h is an infinite history with associated sequence $\{\zeta_n\}_{n=1}^{\infty}$ of LAR's, then $\inf(h) \neq F_i$ if and only if the ζ_n are in G_i infinitely often *or* in R_i only finitely often. The previous lemma shows that acceptance by the family $\mathscr{F}$ is equivalent to the complemented pairs condition $\Omega = \{(G_i, R_i)\}_{i \in I}$ on associated sequences of LAR's. So taking LAR's converts subset acceptance to complemented pairs acceptance, but the co-pairs are now on the memory set of LAR's.[4]

Definition 2.4. Let ζ be an LAR. If ζ displays some sets which are in $\bar{\mathscr{F}}$, then since sets are displayed by suffixes these sets are arranged in descending order

$$F_{i_1} \supset \cdots \supset F_{i_s}.$$

Then the sequence $\sigma = (i_1, \ldots, i_s)$ is the *selection priority* of ζ. If ζ does not display any sets in $\bar{\mathscr{F}}$, let $\sigma = \varepsilon$, representing the empty sequence.

It is important to note that in Definition 2.3, since the ith green set G_i is defined to consist of those LAR's which do *not* display F_i, the selection priority records the indices of those green sets which do not contain ζ.

Any acceptance condition for an automaton can be transformed to an equivalent subset condition without changing the other characteristics (such as the state set and transition function) of the automaton. If we are *given* an alternating automaton M whose acceptance condition is already defined by a set $\Omega = \{G_i, R_i\}_{i \in I}$ of complemented pairs then, at least if the number of co-pairs is relatively small, we should be able to use less memory than the LAR. We turn to the question of defining a suitable "memory" in this case. We define a new type of "appearance record" which is modeled on the LAR but which mainly involves the index set I of Ω. (Compare also [22].)

The important point is that the "index appearance record", or IAR, which we are about to define simply records the current state s of the automaton and a complete list of those indices such that s is *not* in the corresponding green set; but it does so in "LAR

[4]We point out that if we start with the given final family $\mathscr{F} = \{X_1, \ldots, X_r\}$ and now let G_j^* be the set of LAR's which shift X_j and let R_j^* be the set of LAR's which do not display X_j then the acceptance condition is also defined by the set of *direct* pairs $\Omega^* = \{(G_j^*, R_j^*)\}$.

fashion" – old indices are preserved at the left. We shall see why the order is important a little later.

Definition 2.5. Let $\mathscr{Z} = I^{\#} \times Q$, where $I^{\#}$ denotes the set of all sequences without repetition from the set I. The *index appearance record* is the function $\mu: \mathscr{H} \to \mathscr{Z}$ defined as follows. If $q_0 \in G_i$ for all $i \in I$, then $\mu(q_0) = (\varepsilon, q_0)$. Otherwise, $\mu(q_0) = (\sigma_0, q_0)$ where σ_0 is the sequence $(j_1, \ldots, j_c)$ of indices (in numerical order) such that $q_0 \notin G_j$. Suppose that $\mu(q_0\tau_1 \ldots k_n s_n) = (\sigma, s_n)$ has already been defined. Then $\mu(q_0\tau_1 \ldots k_n s_n \tau_{n+1} k_{n+1} s_{n+1})) = (\hat{\sigma}, s_{n+1})$ where $\hat{\sigma}$ is defined in the following way. If $s_{n+1} \in G_i$ for all $i \in I$ then $\hat{\sigma} = \varepsilon$. Otherwise write $\sigma = (i_1, \ldots, i_b)$. Let $\rho = (j_1, \ldots, j_c)$ be the sequence of *all* indices (in numerical order) such that $s_{n+1} \notin G_j$. Remove any i's from σ which do not occur in ρ and let σ' be the resulting sequence. Now remove all indices which occur in σ' from ρ and let ρ' be the resulting sequence. Set $\hat{\sigma} = \sigma'\rho'$. Note that $\hat{\sigma}$ is a permutation of ρ. (For example, if $\sigma = (1, 3, 7, 4)$ and $\rho = (1, 5, 6, 7)$ then $\sigma' = (1, 7)$, $\rho' = (5, 6)$ and $\hat{\sigma} = (1, 7, 5, 6)$.) The component $\hat{\sigma}$ of $(\hat{\sigma}, s)$ is called the *selection priority*. We call $\mu(h)$ the *memory* of h.

We have considered alternating automata whose acceptance condition is either a subset condition or is already a complemented pairs condition. We begin to see that these cases are not really that different. In the first case, the appropriate memory is the LAR which converts subset acceptance to an equivalent complemented pairs condition on the memory set. If we are already given complemented pairs the appropriate memory is the IAR. We now switch to a uniform terminology and notation. In both cases we call the appropriate appearance record the *memory* and use $\mathscr{Z}$ to denote the memory set. We use Ω to denote the set of complemented pairs in both cases. The "sameness" of these cases begins to be revealed in the fact that selection priorities do the same thing in both – they record the indices of the green sets that the memory is currently outside of. Although we shall not see its real significance until the end of the next section, we now pinpoint the crucial property of the selection priorities.

Lemma 2.2 (The Stability Lemma). *Suppose that an infinite play p fails to be winning for OR by the cth complemented pair. Let $\{\zeta_n\}_{n=1}^{\infty}$ be the associated sequence of memories. Then there exists a subscript n_0 and sequence of indices $i_1, \ldots, i_k$ such that for all $n \geqslant n_0$ the selection priority σ_n of ζ_n has the form $\sigma_n = (i_1, \ldots, i_k, c, \ldots)$, where the indices coming after c may still vary according to n.*

Proof. The lemma says that if $n \geqslant n_0$ then the index c occurs in the selection priority and the indices have stabilized down to c. We recall that $\zeta_n \notin G_i$ whenever $i \in \sigma_n$, so those ζ_n with $n \geqslant n_0$ are now permanently outside the green sets for the stable indices $i_1, \ldots, i_k, c$. That *p fails* to win by (G_c, R_c) means that elements of G_c occur only finitely often *and* elements of R_c occur infinitely often. There are two cases, depending on whether we were given subset acceptance or complemented pairs acceptance. First

suppose that we were originally given subset acceptance. Then the memory is the LAR and Lemma 2.1 says that we are supposing that $\inf(p) = F_c \in \bar{\mathscr{F}}$. Choose a subscript n_0 such that all states of F_c have already occurred and just states of F_c will occur after n_0. If $n > n_0$ then only the arrangement of elements of F_c can change in a suffix of ζ_n. Thus, those sets of $\bar{\mathscr{F}}$ which are displayed and which include F_c are now fixed at a sequence

$$F_{i_1} \supset \cdots \supset F_{i_k} \supset F_c.$$

(Those proper subsets of F_c that are displayed and are in $\bar{\mathscr{F}}$, if any, may change.) Also, $\zeta_n \notin G_\ell$, for $\ell = i_1, \ldots, i_k, c$ by definition. So selection priorities have stabilized.

Now suppose that we were given complemented pairs acceptance. Then the memory is the IAR and ζ_n has the form (σ_n, s_n). We can choose a subscript n_c such that $s_n \notin G_c$ for $n \geqslant n_c$. We are now permanently outside G_c so the index c will never be moved in the sequences σ_n. The sequences σ_n may contain indices which occur before the index c. If i is such an index and s_n ever enters the set G_i with $n > n_c$ then i will be removed from its position before c and if later reintroduced will then always occur *after* c. For such an i we can choose $n_i > n_c$ such that i never occurs before c if $n \geqslant n_i$. Let n_0 be the maximum of n_c and all such n_i. Write $\sigma_{n_0} = (i_1, \ldots, i_k, c, \ldots)$. Then if $n \geqslant n_0$, σ_n agrees with σ_{n_0} on all initial components through the one containing c. Now $s_n \notin G_\ell$ for $n \geqslant n_0$ and $\ell = i_1, \ldots, i_k, c$ and the lemma is established. □

This comparison of the LAR and the IAR shows how memory really works from our point of view. If we are given an alternating automaton M, we need a memory function which converts the given acceptance condition to a complemented pairs condition. This condition need not be on the original state set but, more generally, is on the memory set. Then we need to be able to define selection priorities in a stable way. This leads us to an axiomatic definition of a general memory framework.

Definition 2.6. A *memory framework* for an automaton M is a triple $\langle \mu, \Omega, \Pi \rangle$ with the following properties. Let $\mathscr{H}$ be the set of all possible finite histories (initial plays) and let $\mathscr{Z}$ be a finite set of *memories*. Then $\mu: \mathscr{H} \to \mathscr{Z}$ is a function which we assume only satisfies the property that if h and h' are histories with the same memory then $\mu(h\tau kq) = \mu(h'\tau' k' q)$. In other words, μ can be regarded as a function $\mu: \mathscr{Z} \times Q \to \mathscr{Z}$ which depends only on the previous memory and the current state and is thus some kind of general "appearance record". The set $\Omega = \{(G_i, R_i)\}_{i \in I}$ is a set of complemented pairs on the memory set $\mathscr{Z}$. If $p = q_0\tau_1 k_1 s_1 \ldots \tau_n k_n s_n \ldots$ is an infinite play then its *associated memory sequence* is the sequence $\{\zeta_n\}$ where $\zeta_n = \mu(q_0\tau_1 k_1 \ldots \tau_n k_n s_n)$. It is assumed that p is winning for OR if and only if $\{\zeta_n\}$ satisfies the collection Ω of complemented pairs. The *priority function* $\Pi: \mathscr{Z} \to I^*$ (where I^* is the set of sequences without repetition from the index set I of Ω) associates to each memory ζ an ordered list of all the indices i such that $\zeta \notin G_i$. The framework $\langle \mu, \Omega, \Pi \rangle$ is said to be *stable* if it satisfies the following axiom.

The Stability Axiom. If $\{\zeta_n\}_{n=1}^{\infty}$ fails to win according to the cth complemented pair (G_c, R_c) then there exists a subscript n_0 and fixed indices $i_1, \ldots, i_k \in I$ such that if $n > n_0$ then $\Pi(\zeta_n)$ has the form $(i_1, \ldots, i_k, c, j_1, \ldots, j_r)$.

A remark is needed for the complemented pairs case. The axiomatic definition supposes that the complemented pairs condition is on the memory set $\mathscr{Z}$. If we are given M with a set $\Omega = \{(G_i, R_i)\}_{i \in I}$ of complemented pairs, simply define $\Omega' = \{(G_i', R_i')\}_{i \in I}$ where $G_i' = \{(\sigma, s)\colon s \in G_i\}$ and $R_i' = \{(\sigma, s)\colon s \in R_i\}$. This just formally transfers the given acceptance condition to the memory set.

If the memory function can be taken to be simply the last state, we say that we have *zero-memory*. We now show that this happens in the case of an alternating automaton M whose acceptance condition satisfies the *chain condition*: It is a collection of complemented pairs where the green sets from a chain $G_1 \subset \cdots \subset G_m$. (There is *no* hypothesis on the red sets.) Note that the chain condition certainly holds if there is only one co-pair. Given only the last state s, we can calculate the selection priority $\Pi(s) = (1, \ldots, i)$, where i is the largest index such that $s \notin G_i$. (If s is in all the G_i then $\Pi(s) = \varepsilon$.) It is clear that the Stability Axiom holds. If states s_n are permanently outside the green set G_c for $n > n_0$ then they are also outside $G_1, \ldots, G_{c-1}$ since the sets form a chain. (Appendix A shows that a zero-memory strategy need not exist for two complemented pairs when they do not form a chain.)

We always work with complemented pairs. In general, transforming direct pairs into complemented pairs is very complicated. There is, however, an interesting case in which a direct pairs condition can be very simply transformed into a complemented pairs condition.

Definition 2.7. A set $\{(G_i, R_i)\}$, $i = 1, \ldots, n$, of direct or complemented pairs satisfies the *double chain condition* if both the green and red sets form ascending chains $G_1 \subset \cdots \subset G_n$ and $R_1 \subset \cdots \subset R_n$.

Lemma 2.3 (The Shift Lemma). *A direct pairs condition* $\{(G_i, R_i)\}$, $i = 1, \ldots, n$, *which satisfies the double chain condition is equivalent to the complemented pairs condition*

$$\{(\emptyset, R_1), (G_1, R_2), \ldots, (G_{i-1}, R_i), \ldots, (G_{n-1}, R_n), (G_n, \overline{G_n})\},$$

which also satisfies the double chain condition. (The red sets have been "shifted down".)

Proof. First suppose that an infinite sequence of states satisfies the given direct pairs condition. Then there is an index i such that the sequence hits G_i infinitely often and R_i only finitely often. Thus the sequence hits every G_ℓ with $\ell \geqslant i$ infinitely often since the G's form a chain. Also, the sequence hits every R_j with $j \leqslant i$ only finitely often since the R's form a chain. Thus the sequence satisfies the displayed set of complemented pairs.

Note that the displayed set of complemented pairs also satisfies the double chain condition since R_n is included in $\overline{G_n}$. Now suppose that an infinite sequence of states

satisfies the complemented pairs condition. Since the last co-pair is satisfied the sequence must hit G_n infinitely often. (If it hits $\overline{G_n}$ only finitely often it certainly hits G_n infinitely often.) If the sequence does not hit G_{n-1} infinitely often then, since the next co-pair is satisfied, it must hit R_n only finitely often, and thus the direct pair (G_n, R_n) is satisfied.

Now repeat the preceding argument with the next pair. If the sequence hits G_{n-1} infinitely often but not G_{n-2} then the direct pair (G_{n-1}, R_{n-1}) is satisfied. Working our way down the chain, we find a satisfied direct pair (G_j, R_j), $j > 1$, or we arrive at the situation where the sequence hits G_1 infinitely often. Since the first complemented pair must be satisfied and a sequence cannot hit the empty set infinitely often, the sequence must hit R_1 only finitely often and the direct pair (G_1, R_1) is satisfied. □

Thus a collection of *direct* pairs which satisfies the double chain condition also has a memory framework in which the memory is only the last state. In particular, this is the case if there is only one direct pair.

3. Uniformizing strategies: The L-tree construction

This section contains the basic construction of our proof. It can be read in different ways. The reader can choose one of the three cases which we have discussed, that is, to suppose we are given an alternating automaton using subset acceptance, complemented pairs acceptance, or indeed, that we are in the zero-memory case. The word "memory" then refers respectively to the LAR, the IAR, or simply the last state. On the other hand, the reader can forget specific details and verify that the proof uses only the notion of a stable memory framework.

We follow the notation of the previous section: We suppose that we are given an alternating automaton M and a stable memory framework $\langle \mu, \Omega, \Pi \rangle$ for M. We also suppose that we are given an *arbitrary* strategy S for the acceptance game $\Gamma(M, t)$ of M on an input tree t. These data remain fixed. Note that we do *not* assume that S is winning.

Definition 3.1. Let the set of complemented pairs be $\Omega = \{(G_i, R_i)\}_{i \in I}$ where $G_i \cap R_i = \emptyset$. Then we can write the memory set $\mathscr{Z}$ as a disjoint union $\mathscr{Z} = G_i \cup Y_i \cup R_i$, where $Y_i = \mathscr{Z} \setminus (G_i \cup R_i)$. We call this partition of $\mathscr{Z}$ the *ith trichotomy*.

In "color terminology", we partition $\mathscr{Z}$ into green, yellow and red subsets. The acceptance condition is that for *each* i, one must see green elements infinitely often *or* red elements only finitely often.

We give a brief outline of our method before turning to a formal description. By induction on n, we are going to construct an auxiliary ordered tree L_{i,λ_n} for each index $i \in I$ and each path λ_n of length n on the input tree t. The purpose of these trees is to *select* certain histories. Those vertices of each tree L_{i,λ_n} at the maximum level will occur at level n, and will be the only ones which may be extended in the next iteration. They will be called *n-level leaves* and will be labeled by sets of histories of length n. We think of each edge in a tree L_{i,λ_n} as going in one of three directions: left, straight or right, corresponding to the colors green, yellow and red, respectively,.

Given a direction k_{n+1} on t, to continue the path to $\lambda_{n+1} = \lambda_n k_{n+1}$, we calculate the trees $L_{i,\lambda_{n+1}}$ as follows. For each n-level leaf u, find the value of the given strategy S on the histories present in its label U, and see which extensions go in the direction k_{n+1}. Those extensions, if any, which have memories in G_i are put in a single set W labeling a single new vertex which is the left successor of the n-level leaf at the far left side of L_{i,λ_n}. Those histories which extend a history in a label U at an n-level leaf u of L_{i,λ_n} and have memories in Y_i (respectively, R_i) are put in the label of a straight (respectively, right) successor vertex of u. At this step we will have temporary trees with, in general, more than one vertex at level $n+1$ having a label with histories having a given memory ζ. For each such ζ we keep only one history in each final tree $L_{i,\lambda_{n+1}}$. The choice of this history will depend on all trees L_{i,λ_n} in a way determined by the selection priority of the memory ζ. All other histories with memory ζ are discarded from the temporary trees to form the final trees $L_{i,\lambda_{n+1}}$. We now turn to a formal description.

We inductively construct for each finite path λ_n of the input tree t and each index $i \in I$, an auxiliary tree L_{i,λ_n}. Each $L_{i,\varepsilon}$ consists of a single vertex labeled by $\{q_0\}$. We inductively assume:

(1) If $\lambda_n = \lambda_{n-1}k_n$, the initial subtree of L_{i,λ_n} up to level $n-1$ is $L_{i,\lambda_{n-1}}$.
(2) For each i, the labels on n-level leaves in L_{i,λ_n} are pairwise disjoint sets of n-histories whose position is λ_n. Each such history is generated by the strategy S.
(3) For each memory $\zeta \in \mathscr{Z}$ there is at most one n-history with memory ζ which occurs in the labels of n-level leaves in L_{i,λ_n}. If h is a history which occurs in the label on an n-level leaf u, and if $\zeta = \mu(h) \notin G_i$ then h is an extension of a history in the label on the predecessor of u. The leftmost vertex is special. A history which occurs in the label on the leftmost vertex has its memory in G_i and need *not* be an extension of a history on the label on the predecessor.
(4) The union of all the sets labeling n-level leaves in the tree L_{i,λ_n} is the same for all $i \in I$. (The "global" set of histories depends only on λ_n, but histories will be positioned differently in trees with different indices i.)
(5) The width[5] of all the L_{i,λ_n} is bounded by $|\mathscr{Z}| + 1$.

Suppose that all the L_{i,λ_n} have already been constructed and let $\lambda_{n+1} = \lambda_n k_{n+1}$. We construct the trees $L_{i,\lambda_{n+1}}$ in three steps as follows.

[5]The width of a tree is defined as the maximum number of vertices occurring at any level.

Trichotomy step: Let

$(*)_i$ $\quad U_{i,1}, U_{i,2}, \ldots, U_{i,j_i}$

be, in order from left to right, all the *nonempty* sets labeling n-level leaves $u_1, u_2, \ldots, u_{j_i}$ in L_{i,λ_n}. For each history h in such a set, the strategy S determines $S(h)$, which is a term τ_{n+1} of the transition function $\delta(a, q)$ (where a is the letter of the input tree t at λ_n and q is the last state in h). For each generator (k_{n+1}, s_{n+1}) with direction k_{n+1} which occurs in τ_{n+1}, form the extending history $h' = h\tau_{n+1}k_{n+1}s_{n+1}$. (Since M is alternating, one h may have several extensions in the direction k_{n+1}.) For each $j = 1, \ldots, j_i$, let $V_{i,j}$ be the set of all such histories extending some history in $U_{i,j}$.

The set W_i consists of all histories which occur in any of the $V_{i,j}$ and which have memories in G_i. For each j, $W_i(j, Y)$ consists of those histories in $V_{i,j}$ with memories in Y_i, and $W_i(j, R)$ consists of those histories in $V_{i,j}$ with memories in R_i. (Of course, some of these sets may be empty.) We now have an *ordered* sequence of sets

$(**)_i$ $\quad W_i, W_i(1, Y_i), W_i(1, R_i), \ldots, W_i(j_i, Y_i), W_i(j_i, R_i)$.

History selection step: If $\zeta \in \mathscr{Z}$ let $\sigma = (i_1, \ldots, i_s)$ be its selection priority. Recall that σ is a list of the indices such that ζ is outside the corresponding green sets. If σ is empty then ζ is in all the green sets. Select any one history h with memory ζ (provided that one occurs).[6] In every sequence $(**)_i$, leave h in the set in which it occurs and delete all other histories with memory ζ from the sets in which they occur.

If $\sigma = (i_1, \ldots, i_s)$ is nonempty, we shall select a history h with memory ζ which seems the least likely to become winning. Let H_1 be the set of histories with memory ζ which occur as far to the right as possible in the sequence $(**)_{i_1}$. Let H_2 be the subset of H_1 consisting of those histories in H_1 which come from a set as far to the right as possible in $(**)_{i_2}$. Continue in this way until a final subset $H_s \subseteq H_{s-1}$ is obtained. Select one history $h \in H_s$. In *every* sequence $(**)_i$, $i \in I$, leave h in the set in which it occurs and delete all other histories with memory ζ from the sets in which they occur.

Having made a selection for every $\zeta \in \mathscr{Z}$, then for each $i \in I$, we have a final sequence

$(***)_i$ $\quad X_i, X_i(1, Y_i), X_i(1, R_i), \ldots, X_i(j_i, Y_i), X_i(j_i, R_i)$.

Extension step: The $(n+1)$-level leaves in $L_{i,\lambda_{n+1}}$ are defined as follows. The n-level leaf u_j of L_{i,λ_n} labeled by $U_{i,j}$ has a right successor labeled by $X_i(j, R_i)$ if this set is nonempty and a straight successor labeled by $X_i(j, Y_i)$ if this set is nonempty. If both these sets are empty then u_j has no successors in $L_{i,\lambda_{n+1}}$ unless it happens to be the leftmost n-level leaf of L_{i,λ_n} as described next. The leftmost n-level leaf u_0 of L_{i,λ_n} is special and is the only such leaf which has a left successor. The vertex u_0 always has a left successor labeled by X_i even if X_i or the label on u_0 is empty. Each history h' in

[6] When we say "select" a history from a given set, we suppose that this is done according to some definite rule such as taking the history which is least in the lexicographical order. The specific rule is not important, but we want the L-tree construction to be completely deterministic once S is fixed.

a label at level $n+1$ of $L_{i,\lambda_{n+1}}$ must be the extension of a history h in a label at level n of its parent vertex unless $\mu(h')\in G_i$ and h' is thus in the label X_i on the leftmost $(n+1)$-level leaf of $L_{i,\lambda_{n+1}}$.

It is clear that the width of $L_{\lambda_{n+1}}$ is bounded by $|\mathscr{Z}|+1$, and that the other inductive properties hold. This completes the definition of the trees L_{i,λ_n}.

The continuation property (1) ensures that, given an infinite path λ in the input tree t, the *infinite L-tree* $L_{i,\lambda}$ is well-defined, where if $\lambda_n = k_1 \ldots k_n$ is the initial n-segment of λ then $L_{i,\lambda}$ coincides with L_{i,λ_n} up to level n. The properties of these infinite trees $L_{i,\lambda}$ are crucial to our argument. We next turn to a general argument about infinite ordered trees.

Definition 3.2. We consider rooted ordered trees in which each vertex has at most three successors. A vertex may have a left, straight or right successor. Our trees are not full and if a vertex u has only a right successor v we still say that the edge from u to v goes right, etc. We call such trees *positional ternary ordered trees.* A vertex v *branches right* if v is the right successor of its predecessor. (In the case of L-trees, this means that the vertex has a label with histories terminating in *red* states.) A path *branches right* if at least one of its vertices branches right. A collection $\mathscr{C}$ of ordered trees has *good branching* if no tree $T\in\mathscr{C}$ contains a path β which branches right infinitely often.

A sequence $\{v_r\}$ of vertices *moves right* if whenever $j<k$ then:

(i) the level of v_k is the same as the level of v_j and v_k is to the right of v_j, or

(ii) the level of v_k is greater than the level of v_j and either

(iii) v_j and v_k are descendants of vertices u_j and u_k, respectively, at the same level of T and u_k is to the right of u_j, or

(iv) v_k is the descendant of v_j and on the path $v_j,\ldots,v_k$ at least one of the vertices $v_{j+1},\ldots,v_k$ branches right.

Lemma 3.1 (The Branching Lemma). *Let T be a positional ternary ordered tree in which the leftmost vertex at each level has a left successor and is the only vertex of its level having a left successor. Then T has good branching if and only if T does not contain an infinite sequence of vertices which moves right.*

Proof. If T contains a path β which branches right infinitely often then the set $\{v_j\}$ of vertices of β where β branches right is an infinite sequence which moves right. We must show that if T has an infinite sequence $\{v_j\}$ of vertices which moves right then T contains a *path* β which branches right infinitely often. The idea is very simple. At any level ℓ there are only finitely many vertices, say $w_1,\ldots,w_s$, in order from left to right. If $j<k$ and v_j and v_k are two vertices in the sequence which are at level greater than ℓ in the tree, let α_j and α_k be the paths from the origin to v_j and v_k, respectively, and let $w_{r(j)}$ and $w_{r(k)}$ be the vertices at level ℓ through which α_j and α_k respectively pass. Then $r(j)\leqslant r(k)$ because at level ℓ, α_k cannot pass through a vertex which is to the left of the vertex of level ℓ on α_j. Thus as k increases, the position of $w_{r(k)}$ in the

sequence can never move left, and since the number of vertices at level ℓ is finite, from some point onward all paths α_k must pass through the same vertex, say b_ℓ, at level ℓ. The sequence of vertices $b_0 b_1 \ldots b_\ell \ldots$ defines an infinite path β in T.

First of all, the hypothesis on T ensures that the vertices of T which are leftmost at their level all lie along a leftmost path. Furthermore, once a path γ from the origin leaves this leftmost path, it never again contains a vertex which is leftmost at its level. Now, not all the vertices v_j can be on the leftmost path for then the sequence would not move right. If v_j is not leftmost, then v_k is not leftmost for all $k > j$. Thus there exists an index n such that none of the vertices b_ℓ are leftmost for $\ell > n$.

It suffices to show that, for every $\ell > n$, β branches right at some vertex after b_ℓ. Choose indices $j < k$ large enough so that the paths α_j and α_k from the origin to v_j and v_k, respectively, both go through b_ℓ. Then by the definition of "moves right", the path α_k must branch right at some vertex either at or after b_ℓ. If v_k is at level r, then the vertex b_r on β is at least as far to the right as v_k. Thus, β must branch to the right at least once between b_ℓ and b_r. □

We recall that we have not assumed that the strategy S used to construct the L-trees is a winning strategy. We now pinpoint the crucial property possessed by the L-trees if S is winning.

Lemma 3.2. *Let $\mathscr{C} = \{L_{i,\lambda}\}$ be the collection of all ordered L-trees obtained from the strategy S, where i ranges over I and λ ranges over the set of infinite paths on t. If S is a winning strategy, then its collection $\mathscr{C}$ has good branching.*

Proof. In order to prove this lemma by contradiction, suppose that some $L_{i,\lambda}$ contains a path β which branches right infinitely often. After the first time that β branches right no vertex of β is leftmost at its level. (This follows from the property of L-trees that the leftmost vertex always has a left successor and is the only such vertex.) Let b_ℓ be the first vertex of β which is not leftmost. We now construct the "history extension tree" E_β of histories which lie along β. The origin of E_β is a special vertex. At each level $n \geqslant 1$ there is a one-to-one correspondence between the vertices of level n of E_β and the histories present in the label on the vertex $b_{\ell+n-1}$ of β. There is an edge from the origin of β to each vertex at level 1. For $n \geqslant 1$, there is an edge between a vertex u at level n in E_β and a vertex v at level $n+1$ if and only if the history corresponding to v extends the history corresponding to u. By property (3) of the L-tree construction and the fact that u is not leftmost, E_β is indeed a tree. Now E_β is finitely branching and contains arbitrarily long paths, so E_β must contain an infinite path γ by König's Lemma.

The path γ defines an infinite history h. (The initial $(n+\ell)$-segment of h is the history labeling the vertex of level n of γ.) Since no vertex of β is leftmost after b_ℓ, $\mu(h)$ is in G_i only finitely often. Since β branches right infinitely often, $\mu(h)$ hits R_i infinitely often. Thus, h is not accepting according to the complemented pair (G_i, R_i). But h is played by the original strategy S, which contradicts S being a winning strategy. □

We return to the case where S is an arbitrary strategy and use the L-trees $L_{i,\lambda}$ to define a new strategy $\hat{S}$. We call $\hat{S}$ the *L-tree uniformization of S.*

Definition of the Uniform Strategy. We inductively define $\hat{S}$ as follows. First, $\hat{S}(q_0) = S(q_0)$. Now suppose that an initial play $p = q_0\tau_1 k_1 s_1 \ldots k_n s_n$ has been generated by $\hat{S}$. (Note that we do *not* claim p is generated by the strategy S. That is generally false.) Let $\lambda_n = k_1 \ldots k_n$ and construct the L_{i,λ_n} according to the given strategy S. Find the unique history h at level n in L_{i,λ_n} with the same memory ζ as p and define $\hat{S}(p) = S(h)$. (Such an h always exists by induction and the fact that μ depends only on the previous memory and the current state. Since $\hat{S}$ always chooses a term τ chosen by S *on some* history, every generator in τ with a given direction extends at least one history in the L-tree construction. And h is certainly unique since the L-trees select only one history with a given memory.)

First of all, we note that $\hat{S}$ is indeed a strategy since the construction of the L_{i,λ_n} depends only on the path λ_n. Second, it is clear that $\hat{S}(p)$ depends only on the position and the memory of p. Third, it is clear that $\hat{S}$ is a uniformization of S. We now prove that if the collection $\mathscr{C}$ of all infinite L-trees has good branching then the new strategy $\hat{S}$ is winning. We again emphasize that we do *not* assume that the given strategy S itself is winning.

Lemma 3.3. *If the collection $\mathscr{C}$ of all infinite L-trees has good branching then the strategy $\hat{S}$ is winning.*

Proof. Let $p = q_0\tau_1 k_1 s_1 \ldots$ be an infinite play according to the strategy $\hat{S}$ and let $\lambda = k_1 k_2 \ldots$ Let $\zeta = \{\zeta_n\}_{n=1}^{\infty}$ be the associated memory sequence. Now, OR wins unless p fails some complemented pair in Ω. Suppose that p fails the cth complemented pair. Then the Stability Axiom gives a subscript n_0 and a sequence $i_1, \ldots, i_k \in I$ such that for all $n \geqslant n_0$, the selection priority σ_n for ζ_n has the form

$$\sigma_n = (i_1, \ldots, i_k, c, \ldots),$$

where the indices following c in the selection priority may vary with n.

If $n > n_0$, then in the tree $L_{i_1,\lambda}$, the vertex whose label contains some history h_{n+1} with memory ζ_{n+1} will be the successor to the vertex u whose label contains the history h_n with memory ζ_n *unless* there is a history with the same memory which is found at a vertex farther to the right in $L_{i_1,\lambda}$. When this happens the path being followed in $L_{i_1,\lambda}$ is changed. In such a case we say that i_1 is *injured.* If i_1 is injured infinitely often then $L_{i_1,\lambda}$ contains an infinite sequence of vertices which moves right. But this is impossible by the Branching Lemma and the hypothesis.

Thus i_1 can be injured only finitely often and we can choose a subscript n_1 such that i_1 never is injured when $n > n_1$. Similarly, i_2 can be injured only finitely often after n_1 so we can find $n_2 > n_1$ such that i_2 is never injured if $n > n_2$. Working from left to right we can find a subscript n_k so that none of $i_1, \ldots, i_k$ is injured if $n > n_k$.

We keep the same terminology for c itself. Again, c can be injured only finitely often so we find a subscript n_c such that the history with memory ζ_{n+1} is always an extension of the history with memory ζ_n in the tree $L_{c,\lambda}$ if $n > n_c$. But the sequence $\{\zeta_n\}_{n=1}^{\infty}$ hits R_c infinitely often, because p fails the cth complemented pair, so $L_{c,\lambda}$ has a path which branches right infinitely often. This contradiction proves the lemma. □

The reader familiar with finite injury priority arguments (see Soare [19]) will recognize that the proof of Lemma 3.3 fits the same pattern.

Combining the previous lemmas, we have proven the following result.

Theorem 3.1 (The L-tree Theorem). *Let M be an alternating automaton and let $\langle \mu, \Omega, \Pi \rangle$ be a stable memory framework for M. Then M accepts the input t if and only if there exists a first player strategy S for $\Gamma(M,t)$ such that the collection $\mathscr{C}$ of all infinite L-trees constructed from S has good branching.*

We have characterized acceptance by M in terms of whether or not the collection $\mathscr{C}$ has good branching for some strategy S. It is now clear in principle how to simulate M with a nondeterministic automaton N. The automaton N nondeterministically chooses transitions for M for the *bounded* number of copies (histories) present in the L-trees and accepts if and only if the collection $\mathscr{C}$ has good branching. It remains to establish in the next section that a finite automaton can indeed verify good branching. But assuming this for the moment it is clear that, up to its existence, N is actually equivalent to M. If M accepts, then OR has a winning strategy S. If N makes the same choices then the collection $\mathscr{C}$ has good branching so N accepts. If N accepts then N has chosen a collection $\mathscr{C}$ with good branching. Then the strategy $\hat{S}$ defined by $\mathscr{C}$ is winning and M accepts.

Now strategy uniformization is an immediate consequence of Lemmas 3.2 and 3.3. Let S be any winning strategy for OR for $\Gamma(M,t)$. Construct the collection $\mathscr{C}$ of infinite L-trees from S according to $\langle \mu, \Omega, \Pi \rangle$. Then $\mathscr{C}$ has good branching because S is winning. Thus the L-tree uniformization $\hat{S}$ defined by $\mathscr{C}$ is winning. Now, $\hat{S}$ is a uniformization of S and $\hat{S}(h)$ depends only on $\mu(h)$ and the position of h. Thus, we have the following theorem.

Theorem 3.2 (The Strategy Uniformization Theorem). *Let M be an alternating automaton and let $\langle \mu, \Omega, \Pi \rangle$ be a stable memory framework for M. If S is any winning strategy for OR for $\Gamma(M,t)$, then OR has a μ-uniform winning strategy $\hat{S}$ which is a uniformization of S and which is almost finite state in S.*

As mentioned in the introduction, the Duality Principle proved in Appendix C shows that a winning strategy for AND for $\Gamma(M,t)$ directly yields a winning first-player strategy for the dual game $\Gamma(\tilde{M},t)$. There are two cases where the memory

remains the same for the dual game. If, as is always possible for any alternating automaton, M is described so that it uses a subset condition, then the complement is also a subset condition and the memory is the LAR for both games. Suppose that M has an acceptance condition which can be described as a set of complemented pairs satisfying the double chain condition. Then the memory is simply the last state. If AND has the winning strategy, then acceptance in the dual game is the set of direct pairs obtained by interchanging the green and red sets, which thus also satisfies the double chain condition. By Lemma 2.3, the Shift Lemma, we can rewrite this condition as a set of complemented pairs, again still satisfying the double chain condition. We can thus uniformize AND's strategy in $\Gamma(M,t)$ by regarding it as the player in the dual game. If M has an acceptance condition which is a set of direct pairs satisfying the double chain condition, first rewrite this condition as a set of complemented pairs using the shift lemma and then apply the preceding argument. In summary, we have the following theorem.

Theorem 3.3 (Forgetful Determinacy). *Let $\Gamma(M,t)$ be the acceptance game of the alternating automaton M on the input t. One of the players has a winning strategy which depends only on the position on the input tree and the LAR. If M uses a set of either direct or complemented pairs which satisfies the double chain condition then one of the players has a winning strategy which depends only on position and the current state. In any case, if S is any winning strategy then the player with that strategy has a winning strategy $\hat{S}$ which is uniform in the appropriate memory and which is almost finite state in S.*

4. Coding trees

Our motivation is to code the trees L_{i,λ_n} which arose in the previous section into the states of a finite nondeterministic automaton, but details are about coding an arbitrary positional ternary ordered tree of bounded width so we work in that context. Our coding is similar, of course, to that of Safra [17], but we use ternary trees.

Let L be a positional ternary ordered tree of width bounded by a fixed constant w. Let L_n be the initial subtree of L consisting of the vertices and edges of L of depth less than or equal to n. We show how to associate with each L_n a "J-tree" J_n which "codes the branching of L_n" in such a way that the sizes of the J_n are uniformly bounded by $2w$. The object is to construct the J_n so that an automaton can detect whether or not L has good branching by using states which are constructed from the trees J_n.

Let $d = 3w$ and let $V = \{1,\ldots,d\}$ be a set of *names* for vertices of the J_n. At each stage a name will have one of the three colors: red, yellow or green. The initial tree J_0 consists of a single vertex with name (number) 1 and color green. A name which is not attached to a vertex at a given stage n is called *available* and has color green.

A vertex v of L_n is said to be *active* if v lies on a path in L_n from the root to a vertex at the maximum level n. Let A_n be the set of active vertices of L_n and let V_n be the set of all

vertices of J_n. The key to proving the desired properties of the J-trees is to inductively define a function $\phi_n: A_n \to V_n$ at each stage $n = 0, 1, \ldots$ so that certain properties hold. To start the induction, we observe that L_0 just consists of the root a of L, so we let $\phi_0(a) = v$, where v is the single vertex of J_0. Suppose inductively that J_n has already been constructed and we have a surjective function $\phi_n: A_n \to V_n$ such that:

(1) ϕ_n restricted to the n-level vertices in L_n is a one-to-one (left to right) order-preserving correspondence between these active vertices and the terminal vertices of J_n.

(2) The sequence $v_1, \ldots, v_k$ is a path in J_n if and only if $\phi_n^{-1}(\{v_1, \ldots, v_k\})$ is a path in L_n. If v_i comes before v_j in the path, then all vertices of $\phi_n^{-1}(v_i)$ come before all vertices of $\phi_n^{-1}(v_j)$.

(3) If (u, v) is an edge in J_n then the path $\phi_n^{-1}(v)$ branches right at least once if and only if v is either yellow or red.

The J-tree construction: As mentioned before, we inductively assume that J_n and ϕ_n have already been constructed. We construct J_{n+1} in the following four stages.

1. *Copying*: We take an exact copy J'_n of J_n except that vertices which were red in J_n now become yellow in J'_n. Other vertices keep their previous color. Since J'_n is a copy of J_n we can regard ϕ_n as mapping from A_n to the vertices of J'_n. The three properties of ϕ_n are clearly preserved.

2. *Extension*: Now ϕ_n is a one-to-one correspondence between n-level vertices in L_n and the terminal vertices of J'_n. All $(n+1)$-level vertices in L_{n+1} are successors of vertices of level n. Always taking the name available with the least number as the name for each of the new vertices, and working from left to right, extend each terminal vertex of J'_n in exactly the same way that the corresponding vertex of L_n is extended in L_{n+1}. New vertices are colored green except that right successors are colored red. Let the resulting tree be J''_{n+1}. Now ϕ_n clearly extends to a new function, which we shall call $\hat{\phi}_{n+1}$, from $A_n \cup A_{n+1}$ to the vertices of J''_{n+1} and vertices of level $n+1$ in L_{n+1} map to these new vertices. Properties 2 and 3 are clearly satisfied by the extension. There may now be terminal vertices of J''_{n+1} which correspond to vertices of level n which were not extended to level $n+1$ in L_{n+1}. We remove these vertices in the next step.

3. *Deletion*: If v is a vertex of J''_{n+1} which is terminal but which does not correspond to a vertex of level $n+1$ in L_{n+1}, then delete v and the edge incident to v. The name labeling v is made available and is colored green. This alters the tree J''_{n+1} to form a new tree. The function $\hat{\phi}_{n+1}$ is then modified so that the vertices in the preimage of v are deleted from its domain. This operation is repeated until it cannot be applied to any vertices. If v is deleted then no vertices in the former preimage of v under $\hat{\phi}_{n+1}$ are active at stage $n+1$ and when no further deletions are applicable $\hat{\phi}_{n+1}$ maps the set A_{n+1} of active vertices at stage $n+1$ onto a final tree which we shall call J^*_{n+1}.

4. *Contraction*: We must now keep the J-tree from becoming too large by contracting interior vertices of degree two. Suppose that u, v and w are vertices of the current J-tree J^*_{n+1} with v a successor of u, and w the unique successor of v. (Note that w may have one or more successors.) In this case we *contract* the edge vw to the vertex v,

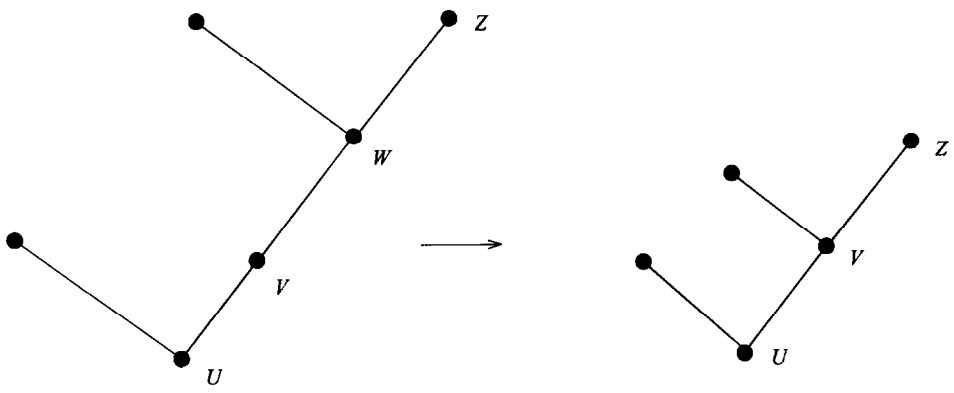

Fig. 1.

identifying w with v as in Fig. 1. The successors of w are now successors of v. We modify the vertex correspondence by defining ϕ_{n+1} to be the same as $\hat{\phi}_{n+1}$ except that $\phi_{n+1}(a) = v$ for all vertices a which were mapped to w by $\hat{\phi}_{n+1}$. The name which labeled w is made available. If w was yellow or red then v is made red. If w was green then v keeps the color it had before contraction. Contraction is applied until it is not applicable, obtaining a final tree J_{n+1}.

We must verify that the three correspondence properties remain true for ϕ_{n+1}. If w was a terminal vertex in J^*_{n+1} then its predecessor v was not and there was a unique vertex a of level $n+1$ in L_{n+1} with $\hat{\phi}_{n+1}(a) = w$. After contraction v is terminal and a is the unique vertex of level $n+1$ with $\phi_{n+1}(a) = v$.

The second property is also clear since if $v_1, \ldots, u, v, \ldots$ is a path after contraction and $v_1, \ldots, u, v, w, \ldots$ was a path before contraction then

$$\phi^{-1}_{n+1}(\{v_1, \ldots, u, v, \ldots\}) = \hat{\phi}^{-1}_{n+1}(\{v_1, \ldots, u, v, w, \ldots\}).$$

Now $\alpha = \phi^{-1}_{n+1}(v)$ has the form $\alpha_1\alpha_2$ where $\alpha_1 = \hat{\phi}^{-1}_{n+1}(v)$ and $\alpha_2 = \hat{\phi}^{-1}_{n+1}(w)$. By the color rules, v is yellow or red after the contraction if and only w was yellow or red or if v itself was yellow or red before the contraction. In the first case α_2 branches right at least once and in the second case α_1 branches right by the induction hypothesis on ϕ_n. The converse also follows since α branches right at least once if and only if either α_1 or α_2 branches right at least once.

It is clear from the descriptions of the processes of deletion and contraction that vertices $v \in A_n \cap A_{n+1}$, i.e. those which remain active in going from L_n to L_{n+1}, have images $\phi_{n+1}(v)$ under ϕ_{n+1} which are defined as being the same as they have under ϕ_n. Such vertices $\phi_n(v)$ and $\phi_{n+1}(v)$ are then defined as being *retained* in going from the tree J_n to the tree J_{n+1}.

This concludes the description of the construction of J_{n+1} and the verification that the three properties hold. It remains only to verify that the size of J_{n+1} remains bounded by $2w$. Since L_{n+1} has width at most w there are not more than w terminal

vertices of J_{n+1}. Now each internal vertex of J_{n+1} has outdegree at least two so there are at most $w/2$ predecessors of terminal vertices. Continuing in this way we see that J_{n+1} has at most $w(1 + \frac{1}{2} + \frac{1}{4} + \cdots) = 2w$ vertices.

We can now prove the desired result.

Lemma 4.1 (The Coding Lemma). *The tree L has good branching if and only if there is no vertex name v which is red infinitely often and green only finitely often in the sequence (J_n), $n \geqslant 0$.*

Proof. First suppose that some vertex v which is retained is red infinitely often and green only finitely often. Then v must always appear in some tree J_{n_0} and in all subsequent trees from stage n_0 onwards. After stage n_0, the only way that v can become red is to have a yellow or red vertex contracted to it. Thus there are infinitely many stages n_j after n_0 at which a vertex w_j is contracted to v. Define β_0 to be the path from the root of L to the last vertex of $\phi_{n_0}^{-1}(v)$ at stage n_0. Define $\beta_j = \hat{\phi}_{n_j}^{-1}(w_j)$ just before w_j is contracted to v (the actual contraction occurs when ϕ_{n_j} is formed from $\hat{\phi}_{n_j}$). Then $\beta = \beta_0\beta_1\beta_2 \ldots$ is an infinite path in L and infinitely many of the β_j branch right at least once since infinitely many of the w_j are yellow or red at least once.

Conversely, let β be a path in L which branches right infinitely often. Let v be a vertex of as great a depth as possible which is always retained in J_n from some stage n_0 onwards and which has $f^{-1}(v)$ contained in β. Such a v exists since the root 1 is on β and all the maximum depths of all the J_n are bounded. It must be the case that there are infinitely many stages n_j with vertices w_j which are contracted to v and with $\phi_{n_j}^{-1}(w_j)$ contained in β. For, if not, then v would have to have a successor which was retained in the trees J_n from some point onwards and which lay along β. This contradicts the statement that v was chosen so as to have maximal depth. Thus, we can factor β as $\beta_0\beta_1\beta_2\cdots$, where $\beta_j = \hat{\phi}_{n_j}^{-1}(w_j)$, just before w_j is contracted to v. Since β branches right infinitely often, infinitely many of the β_j branch right at least once by property (3) and each such corresponding w_j is yellow or red. Thus, v turns red infinitely often. Also, v is never green after the first time when it becomes red. □

5. Constructing automata

We can now explicitly construct a nondeterministic automaton N equivalent to a given alternating automaton M and thus prove the Simulation Theorem.

Now, the construction of the L-trees $L_{i,\lambda_{n+1}}$ from the L-trees L_{i,λ_n} depends only on the ordered sequences of sets of memories labeling terminal vertices at the final level n of the trees L_{i,λ_n}. We have seen that instead of keeping track of the entire L-trees L_{i,λ_n} it suffices to keep track of the corresponding J-trees J_{i,λ_n} to detect branching. The state set $\mathscr{J}$ of N consists of m-tuples of possible J-trees (for the given width $w = |\mathscr{Z}| + 1$) where the terminal vertices are also labeled by pairwise disjoint subsets

of the memory set $\mathscr{Z}$. Thus at step n, the state of N is the m-tuple $\langle J_{1,\lambda_n}, J_{2,\lambda_n}, \ldots, J_{m,\lambda_n}\rangle$.

We now count the number of possible trees. For each $i \in I = \{1, 2, \ldots, m\}$ we can think of such a tree as specified by a triple $\langle f_i, g_i, h_i\rangle$ of functions which are defined as follows. Take V_i to be the set of all numbers used as vertex names for the ith J-tree. (The V_i are all identical alphabets but are thought of as pairwise disjoint.) Then $|V_i| = 3w$. The function $f_i: V_i \to V_i \cup \{*\}$ gives the current structure of the ith tree. If a name v is not in the tree then $f(v) = *$. If v is in the tree and is not the root, then $f(v)$ is the predecessor of v in the tree. Since the root is always denoted by 1, let $f(1) = 1$. The total number of such functions is bounded by $(3w+1)^{3w} = 2^{O(w \log(w))} = 2^{O(|\mathscr{Z}| \log |\mathscr{Z}|)}$. The function g_i is the color function with $g_i(v)$ the current color of v. The total number of such g_i is $3^{3w} = 2^{O(w)}$. Finally, $h_i: \mathscr{Z} \to V \cup \{*\}$ is the labeling function for terminal vertices of the ith tree. If ζ is not in the label on any terminal vertex then $h(\zeta) = *$. If ζ is in the label of the terminal vertex u then $h_i(\zeta) = u$. Specifying the labeling this way is possible since the sets labeling terminal vertices are pairwise disjoint. The number of such functions h is bounded by $|\mathscr{Z}|^{3w+1} = 2^{O(|\mathscr{Z}| \log |\mathscr{Z}|)}$. Thus the total number of the possible trees in $\mathscr{J}$ is $2^{O(|\mathscr{Z}| \log |\mathscr{Z}|)}$. We have one J-tree for each complemented pair in Ω. Thus, if $|\Omega| = m$ then $|N| \leqslant 2^{bm|\mathscr{Z}| \log |\mathscr{Z}|}$, where b is a constant which is independent of the size of the memory and the number of complemented pairs.

We recall that since N is a nondeterministic automaton, its transition function is a mapping of the form:

$$\eta: \Sigma \times \mathscr{J} \to \mathscr{P}(\mathscr{J} \times \mathscr{J}).$$

It is clear how the transition function works. The initial state of N is the m-tuple $\langle J_{1,0}, \ldots, J_{m,0}\rangle$ where each $J_{i,0}$ consists of a single vertex 1_i, colored green and labeled by $\{q_0\}$. If the current state is $\langle J_{1,n}, \ldots, J_{m,n}\rangle \in \mathscr{J}$ and the current input letter is a, then N can make a possible nondeterministic choice for the transition function of M for the memory (which includes the last state) labeling a terminal vertex of J. Note that if M is deterministic then there is no choice at this point and N will be deterministic since the further description of the transition function is deterministic. For each direction k_{n+1}, calculate the extension trees of the $J_{i,n}$ for the direction k_{n+1} according to the L-tree construction. (Recall that the extension depends only on the sets labeling the vertices at the last level, so this calculation is determined.) Now apply the J-tree rules to find the final new trees in the direction k_{n+1}. The value of η in the direction k_{n+1} is the tuple describing the trees.

The acceptance condition is the set of complemented pairs

$$\Delta = \{(\hat{G}_{i,j}, \hat{R}_{i,j})\}, \quad i \in I,\ j \in V_i,$$

where $\hat{G}_{i,j}$ consists of those tuples in which vertex j of V_i is green and $\hat{R}_{i,j}$ consists of those tuples in which vertex j of V_i is red. Thus, Δ forces every vertex name to be green infinitely often or red only finitely often. By the Coding Lemma, N accepts if and only if

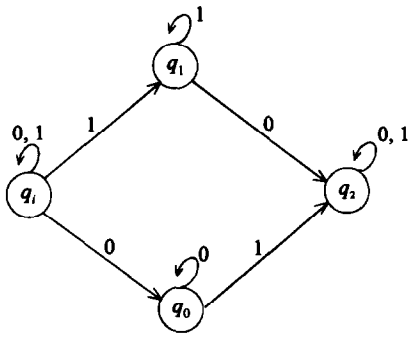

Fig. 2.

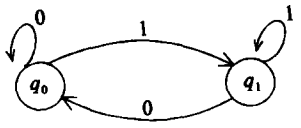

Fig. 3.

the collection $\mathscr{C}$ of L-trees corresponding to the choices of N has good branching. Thus N is equivalent to M by the discussion at the end of the preceding section.

Suppose that we are given a set Ω defining the acceptance condition of M with $m = |I|$ complemented pairs. The memory set $\mathscr{Z}$ is $I^* \times Q$. Now, $|I^*|$ is certainly bounded by $((m+1)!)^2$. Thus we can write $|N| \leqslant 2^{d|M|\log|M|}$, where d depends only on m and is independent of the number of states. Of course, d grows very quickly as m increases. For a set Ω satisfying the chain condition we have $|N| \leqslant 2^{bm|M|\log|M|}$.

Having said this, we consider an example. We work on the line labeled from the alphabet $\Sigma = \{0, 1\}$. Consider the nondeterministic Büchi automaton B which accepts the language K consisting of those sequences containing only finitely many 1's or containing only finitely many 0's, and which has the transition diagram shown in Fig. 2.

B has initial state q_i and $F = \{q_0, q_1\}$. When B is in state q_i it can guess that it has seen the last 0 (on input 1) or that it has seen the last 1 (on input 0) going respectively to q_1 or q_0. If B ever sees the wrong input in either q_0 or q_1 it goes to the rejecting state q_2 and remains there. From the standpoint of Büchi acceptance, B is highly nondeterministic.

The connection between determinization and more powerful acceptance conditions is clearly illustrated by B. If we allow two pairs, then K is accepted by the obvious deterministic automaton shown in Fig. 3 with pairs

$$\{(\{q_0\},\{q_1\}),(\{q_1\},\{q_0\})\}.$$

However, let us consider the dual $\tilde{B}$ of B and see how it works. Now $\tilde{B}$ has transition function $\tilde{\delta}$ defined by

$$\tilde{\delta}(0,q_i) = q_i \wedge q_0, \quad \tilde{\delta}(1,q_i) = q_i \wedge q_1, \quad \tilde{\delta}(0,q_0) = q_0, \quad \tilde{\delta}(1,q_0) = q_2,$$
$$\tilde{\delta}(1,q_1) = q_1, \quad \tilde{\delta}(0,q_1) = q_2, \quad \tilde{\delta}(-,q_2) = q_2.$$

The acceptance condition of $\tilde{B}$ is the complement of that of B, namely the co-Büchi condition defined by the complemented pair $(\emptyset, \{q_0, q_1\})$. It may seem surprising that

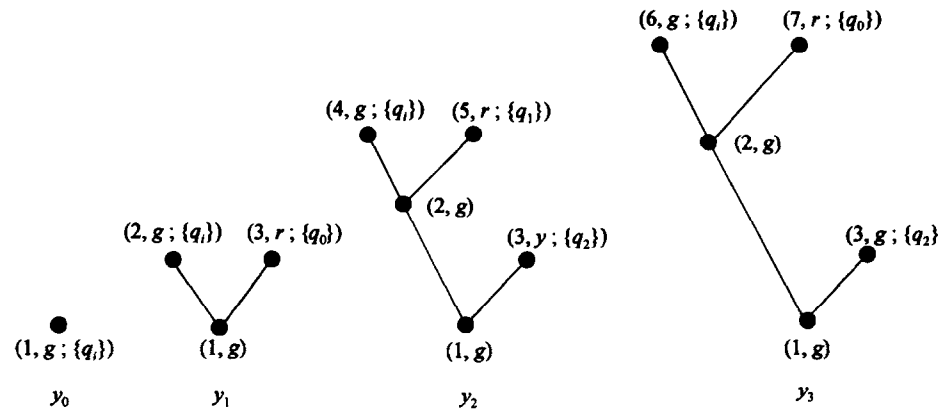

Fig. 4.

$\tilde{B}$ can accept $\bar{K}$, the set of sequences with infinitely many occurrences of both 0 and 1 by such a weak condition. This is a good illustration of how alternating automata can work. There is always a copy of the automaton in the state q_i. Think of this as the "master control". On reading each input, the control starts a copy in q_0 if the input is 0 or starts a copy in q_1 if the input is 1. Call these copies "verifiers". A verifier is "discharged" when it goes to q_2 by reading a letter which is the opposite of its type. The acceptance condition requires all verifiers to be discharged, that is, some time after each input the opposite letter must occur. Thus $\tilde{B}$ accepts $\bar{K}$. As explained in the discussion of McNaughton's Theorem in the introduction, $\tilde{B}$ is a deterministic alternating automaton so N will be deterministic. We apply our construction rules to calculate the behavior of the simulating automaton N on the input $(01)^\omega$, which it accepts, and 0^ω, which is rejected. The fundamental trichotomy for $\tilde{B}$ is $G = \emptyset$, $Y = \{q_i, q_2\}$, $R = \{q_0, q_1\}$. Since G is empty, left successors carry no information and we simply suppress them. The states encountered in reading $(01)^\omega$ are illustrated in Fig. 4.

The label on a vertex consists of its number, color, and labeling states, if any. The sequence of state transitions of N on reading $(01)^\omega$ is

$$y_0 \xrightarrow{0} y_1 \xrightarrow{1} y_2 \xrightarrow{0} y_3 \xrightarrow{1} y_2 \xrightarrow{0} y_3 \xrightarrow{1} \cdots$$

Now vertex numbers 4 and 5 do not appear in alternate stages so they are green infinitely often. Similarly with 6 and 7. So N accepts as it should.

The transition on 0^ω is simply

$$y_0 \xrightarrow{0} y_1 \xrightarrow{0} y_1 \xrightarrow{0} y_1 \xrightarrow{0} \cdots$$

The vertex 3 now remains red permanently so N rejects.

Appendix A. Zero-memory is false for automata with two complemented pairs

In this appendix we give a simple example showing that an automaton whose acceptance condition is defined by two complemented pairs may accept but need not have any zero-memory strategy for acceptance. Our automaton M works on the line with a one-letter alphabet and has the following transition function. (We have suppressed the direction and the input letter since these are unique.) See Fig. 5 below.

$$\delta = (q_0) = q_1 \wedge q_2, \quad \delta(q_1) = q_3 = \delta(q_2), \quad \delta(q_3) = q_4 \vee q_5, \quad \delta(q_5) = q_0 = \delta(q_4)$$

Let acceptance be defined by the complemented pairs condition

$$\Omega = \{(\{q_1\},\{q_5\}),(\{q_2\},\{q_4\})\}.$$

The intuitive idea is that M must "keep left" or "keep right" in the transition diagram in order to accept. Now M goes to both q_1 and q_2 after q_0 and then to q_3 from both states. If M follows a zero-memory strategy then all histories go to the same state after q_3. Since the strategy may depend on position, the state chosen can change each time through q_3, but is always either q_4 or q_5. There are two cases. First suppose that q_5 is chosen infinitely often. Consider any history in which q_1 occurs only finitely often. Such a history fails to accept by the first pair. If the first case does not apply, then q_4 is certainly chosen infinitely often. Consider any history which contains q_2 only finitely often. Such a history fails to accept by the second pair. Thus M cannot accept by any zero-memory strategy.

However, M does have a simple strategy for acceptance: "When in q_3, choose q_5 if the previous state was q_1 and choose q_4 if the previous state was q_2." Now, if a history contains q_1 only finitely often it must contain q_5 only finitely often. Similarly with q_2 and q_4. Thus M accepts according to this strategy.

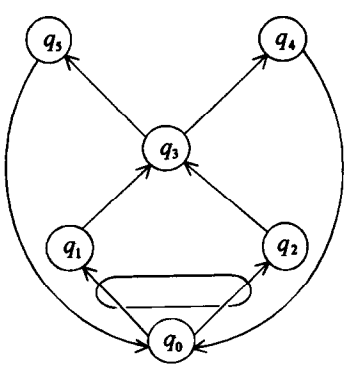

Fig. 5.

Appendix B. Nondeterminism, alternation and covering trees

There is a way of comparing alternation and nondeterminism which may help clarify the relationship between our method and the Gurevich–Harrington Theorem. Let t_0 be the unlabeled k-ary tree with direction set $K = \{0, 1, \ldots, k-1\}$. Let $n \geqslant 1$ be a positive integer and let $\hat{t}_0$ be the unlabeled (nk)-ary tree with direction set $C = \{(0,1), \ldots, (0,n), \ldots, (k-1,1), \ldots, (k-1,n)\}$. Define the *projection map* $\pi : \hat{t}_0 \to t_0$ by

$$\pi((k_1, n_1) \ldots (k_r, n_r)) = k_1 \ldots k_r.$$

The map π simply erases the second component of a path in $\hat{t}_0$. We call t_0 the *base* and $\hat{t}_0$ an *n-fold branched cover* of t_0. If v is a vertex of level ℓ in t_0 then there are n^ℓ vertices u in $\hat{t}_0$ with $\pi(u) = v$. Now consider a k-ary tree t and an (nk)-ary tree $\hat{t}$, both labeled from the alphabet Σ. We say that $\hat{t}$ is a *covering tree of* t if each vertex $u \in \hat{t}$ is labeled by the same letter as $\pi(u)$.

We can now easily *unfold* an alternating automaton M working on t to a nondeterministic automaton N working on an n-fold branched covering tree $\hat{t}$ where $n = |M|$. The state set of N is $Q \cup \{*\}$, where Q is the state set of M and $*$ is a new state indicating the absence of a copy of M. The idea is that $\hat{t}$ has enough directions so that distinct copies of M can go in distinct directions. Precisely, if M has $\delta(a, q) = \tau_1 \vee \cdots \vee \tau_k$ then N has $\hat{\delta}(a, q) = \{\hat{\tau}_1, \ldots, \hat{\tau}_k\}$ where each $\hat{\tau}_i$ is an (nk)-tuple which has q_j in the (d, j)-slot if (d, q_j) appears in τ_i. If τ_i does not send a copy in state q_j in the direction d then $*$ appears in the (d, j)-slot of $\hat{\tau}_0$. Also, on any letter a, $\hat{\delta}(a, *)$ is the single tuple with $*$ in all positions. The acceptance condition of N is the same as that of M with the additional proviso that a history which is $*$ from some point onwards accepts. It is clear that M accepts t if and only if N accepts $\hat{t}$. So an alternating automaton is equivalent to a nondeterministic automaton working on a bigger tree.

We point out that the opposite process is also possible. A nondeterministic automaton N on $\hat{t}$ can be *folded* to an alternating automaton on the smaller tree t. This is the argument for the emptiness problem given in the Introduction: A nondeterministic automaton on a tree labeled by a single letter can be folded to an alternating automaton on the line.

The question of which subtrees of t are isomorphic is very complicated. But the question of whether or not two vertices of $\hat{t}$ have the *same projection* is so simple that a finite automaton can decide the question. We imagine a finite automaton A reading ordered pairs of directions on $\hat{t}$. Given a sequence

$$\langle (k_1, n_1), (k'_1, n'_1) \rangle, \ldots, \langle (k_r, n_r), (k'_r, n'_r) \rangle,$$

the paths $(k_1, n_1) \ldots (k_r, n_r)$ and $(k'_1, n'_1 \ldots (k'_r, n'_r)$ have the same projection exactly if $k_i = k'_i$, $i = 1, \ldots, r$. By the definition of covering tree, if $\pi(u) = \pi(u')$ then the subtrees of $\hat{t}$ beginning at u and u' are isomorphic. "Having the same projection" is only the

"automata level" of the isomorphism question, but reasonably enough, this level suffices for simulating alternating automata.

One can view our proof of strategy uniformization as showing that if a nondeterministic automaton N on $\hat{t}$ (now *not* thought of as coming from an alternating automaton) has a winning strategy S for aceptance then it has a winning strategy $\hat{S}$ in which two copies which are at vertices with the same projection and which have the same memory, make the same transition. On the one hand, this is very much weaker than the Gurevich–Harrington Theorem. We have uniformized only over vertices having the same projection. On the other hand, it is the weak case which allows our extremely strong relative effectiveness conclusion on $\hat{S}$. We have shown that in the projection case, $\hat{S}$ is a uniformization of the given S and is not only effectively calculable from S but is "almost" calculable by a finite automaton. In particular, if S itself is finite state (that is, defined by a finite automaton reading histories) then $\hat{S}$ is also finite state.

Appendix C. Review of alternating tree automata and the form of the transition function does not matter

The automata which we shall discuss work on vertex-labeled, full, infinite, ordered k-ary trees. The vertices of such a tree may be associated with their corresponding paths from the root which are finite strings on the alphabet $K = \{0, 1, \ldots, k-1\}$. Given a finite labeling alphabet Σ, a labeled tree t can therefore be treated as a mapping from this set K^* of finite strings to Σ. A nondeterministic automaton may be thought of as a machine which starts at the root ε of the tree, and repeatedly splits into k copies of itself at each vertex as it follows the infinite paths.

An alternating automaton M is specified by giving five items. They are: two finite sets Q and Σ called the *state set* and the *labeling alphabet*, respectively, a *transition function* δ used to determine the sequence of states through which the automaton progresses (to be described later), an *initial state* $q_0 \in Q$, and an *acceptance condition* which for purposes of this discussion we shall take as subset acceptance so that it consists of a collection $\mathscr{F}$ of subsets of Q. The acceptance condition could, however, equally well be any of the others mentioned in the introduction. Therefore, we write an automaton M as a 5-tuple

$$M = \langle Q, \Sigma, \delta, q_0, \mathscr{F} \rangle.$$

To define the transition function, we must use the concept of the *free distributive lattice* $\mathscr{L}(X)$ on some set X of generators. We define $\mathscr{L}(X)$ as the set of equivalence classes of logical formulas on variables taken from the set X using the operations $\vee$ and $\wedge$ (i.e. disjunction and conjunction) but not negation, up to logical equivalence. These equivalence classes are thus the lattice elements of $\mathscr{L}(X)$ and may be combined in the usual way using the two operations. For each lattice element

$\alpha \in \mathscr{L}(X)$ there is a subset $X(\alpha) \subseteq X$ of the generators upon which the truth value of α depends. For example, the lattice element represented by the formula $x_1 \wedge (x_2 \vee x_3)$ has $\{x_1, x_2, x_3\}$ for its set of generators, but a formula such as $x_1 \wedge (x_2 \vee x_1)$ represents a lattice element which can also be represented by the formula x_1 using fewer variables and hence has $\{x_1\}$ for its set. We list a few elementary results about $\mathscr{L}(X)$.

Proposition C1. *Each lattice element α of $\mathscr{L}(X)$ can be represented uniquely in* irredundant disjunctive normal form *(IDNF). To describe this form, we must specify a collection $\mathscr{T}(\alpha)$ of* defining sets $\tau \subseteq X$, *such that no defining set of $\mathscr{T}(\alpha)$ is properly included in another, i.e. they are incomparable. Then the IDNF of α is*

$$\bigvee_{\tau \in \mathscr{T}(\alpha)} \left(\bigwedge_{x \in \tau} x \right).$$

Proposition C2. *There is a one-to-one correspondence between collections of incomparable subsets of X and lattice elements of $\mathscr{L}(X)$.*

Proposition C3. *The ordering relationship $\alpha \leqslant \beta$ between two lattice elements is equivalent to the relationship*

$$\forall \tau \in \mathscr{T}(\alpha), \quad \exists \chi \in \mathscr{T}(\beta), \ \tau \supseteq \chi$$

between the collections of defining sets which determine the irredundant disjunctive forms.

There are two collections of defining sets which have special properties. These are (i) the empty collection $\mathscr{T}(F) = \emptyset$ and (ii) the collection $\mathscr{T}(T) = \{\emptyset\}$ consisting of just the empty set. They are sets corresponding to minimum and maximum elements lattice elements of $\mathscr{L}(X)$, and are represented by the logical expressions F (false) and T (true) respectively, which are also taken to be their IDNFs.

The present theory can be developed either including these two elements in $\mathscr{L}(X)$ or not, since $\mathscr{L}(X)$ remains a lattice if they are removed. In the body of the paper we hae assumed they are not, and when they are *not*, Proposition C2 above must be modified to specify only nonempty collections of nonempty subsets. In our present development, we assume F and T are present in $\mathscr{L}(X)$. A second difference between our present development and that in the main body of the paper is in our definition of the notion of "play" as noted below.

We have noted in Sections 1 and 2 that the transition function δ is defined as a mapping $\delta: \Sigma \times Q \to \mathscr{L}(K \times Q)$, where δ is represented in IDNF $\phi(\delta)$. There we described how it is used to play a game $\Gamma(M, t)$ between the two players OR and AND on the labeled input tree $t: K^* \to \Sigma$. Starting at the origin ε in the state q_0 on t we saw how at each step n of the game OR chose a defining set (term) τ_{n+1} of $\phi(\delta(a_n, s_n))$, and

then AND chose a generator (k_{n+1}, s_{n+1}) of τ_{n+1}. After step n the *initial play*[7] consists of the sequence $p_n = q_0 k_1 s_1 \ldots k_n s_n$. If we allow $\phi(\delta(a_n, s_n))$ to be the either of the logical elements F and T, then OR may encounter $\phi(\delta(a_n, s_n)) = F$ and so be unable to choose a defining set since none exists. This stops the game and corresponds to a *loss* by OR. Furthermore, AND may be unable to choose a variable in the defining set, if $\phi(\delta(a_n, s_n)) = T$ because the (only) defining set is empty, which also stops the game and corresponds to a loss by AND and therefore a *win* by OR. If $\phi(\delta(a_n, s_n))$ is neither F nor T at any step, then the original rules will be followed for infinitely many steps, with the result that the *infinite play* $p = q_0 k_1 s_1 \ldots$ is *won by* OR (using subset acceptance) if the set $\inf(p)$ of those states appearing infinitely often in p is one of the sets in $\mathscr{F}$. Otherwise, it is *won* by AND, and *lost* by OR. The set of all plays p of M on an input tree t may be arranged as a tree $T(M,t)$. In it each vertex is represented by an initial play p_n. The children of this vertex are all sequences of the form $p_n(k_{n+1}, s_{n+1})$ such that (k_{n+1}, s_{n+1}) is a generator of $\delta(a_n, s_n)$. To each vertex p_n of the tree $T(M,t)$ there corresponds a vertex $k_1 k_2 \ldots k_n$ of the input tree t. The tree $T(M,t)$ may have both finite and infinite paths (plays) if F and T are among the expressions represented by $\delta(a_n, s_n)$, otherwise then only infinite paths will be present.

As explained in Section 2, a *strategy* S for OR is a rule which determines a choice for OR of the defining set τ_{n+1} at each step of the game, based upon the initial play p_n up to the point in question and upon the tree t. Such a strategy S may be thought of as corresponding to a "pruned" version of the tree $T(M,t)$, called the *strategy tree* $T_S(M,t)$, in which only those plays which follow the strategy S are represented.

An alternative way to play this game is to use an arbitrary logical formula, say d_n, possibly not written in IDNF, to represent $\delta(a_n, s_n)$. Then the formula d_n corresponds in the usual way to a finite rooted tree $t(d_n)$ whose leaves are labeled with variables from $K \times Q$, or possibly F or T, and whose internal vertices are labeled with the logical symbols $\vee$ and $\wedge$. The nth step may now be thought of as a finite subgame Γ_n played on $t(d_n)$. Starting at the root, a path is drawn on the tree to one of the leaves, the direction of the path being chosen by either OR or AND at each internal vertex, depending on whether the label at that vertex is $\vee$ or $\wedge$, respectively. The label of the corresponding leaf is then the selected variable for the given step.

A *local strategy* S_n for OR in this finite subgame on $t(d_n)$ consists of the selection of a child vertex on the tree for each vertex with the label $\vee$. This selection determines a "pruned" tree whose leaves are labeled with the generators which form a subset τ' of te variables appearing in d_n. We note that if d_n is already the IDNF for $\delta(a_n, s_n)$ then the local strategy for the subgame is the same as the one we have described earlier.

The overall strategy S for OR then consists of the set of all finite local strateges S_n which are used in the resulting subgames. Since each such local strategy either

[7]The definition for "play" given here differs from that of the main body of the paper in that the terms τ_i chosen by OR are omitted. A consequence of the logical equivalence theorem proved here is that the τ_i terms can never affect any winning strategy of either player and so need never be included in a memory required for such a strategy.

terminates the path in F or T, or determines a subset τ' of the variables appearing in d_n, these may be used to form an overall strategy tree $T_S(M,t)$ as before.

Lemma C1. *Given a representation d of some lattice element $\alpha \in \mathscr{L}(X)$ then* (i), *for any defining set τ of α, there is always a local strategy S for OR on d which yields τ. Also,* (ii), *if S' is any local strategy for* OR *on d the corresponding subset τ' always includes some defining set τ of α.*

Proof. An elementary argument using induction on the depth of the tree $t(d)$ of d proves the lemma. We note that if the representation d for α is in IDNF then d is a disjunction of products of the members of the defining sets, so the local strategy for OR consists of just the choice of a defining set, and the result holds in this case. If the tree $t(d)$ has depth one (or less as in the case of F or T), then the representation d is already in IDNF, and the resulit is true.

We therefore assume, inductively, that the tree $t(d)$ has some depth $n > 1$, and that the result holds if the depth of the tree is less than n. If the label of the root is " $\vee$ ", then the local strategy of OR calls for choosing one of the subtrees of the root on which to play the game. Each defining set of α is a defining set of the lattice element corresponding to one or more of these subtrees. By the inductive assumption, the result holds on this subtree, so property (i) clearly holds, and there is a defining set τ_1 of this subtree which must be included in the set τ' described in (ii). Since τ_1 must include some defining set of α (i.e. of the entire tree), the result follows in this case.

If the label of the root is $\wedge$ then the local strategy of OR is described by a "substrategy" on each of the subtrees of the root. To prove (i), we note that each defining set τ of α is a union of defining sets for the subtrees. Also, to prove (ii), we see that the set τ' is the union of corresponding sets for the various substrategies. Now, each of these sets includes a defining set of the subtree, by the inductive hypothesis. But their union must include a defining set of α, so the result also holds in this case and the inductive step is complete. □

If the steps of an infinite game are played on general representations for the transition function $\delta(a_n, q_n)$ which are not necessarily in IDNF, then by the first part of the lemma, we see that OR may use the same overall strategy that it could have used if the representations had all been in IDNF. Therefore, any strategy S for the infinite game that could be followed in the latter case could also be followed in the former. This means that if the input tree t is accepted by M when the formulas are always in IDNF, then it is also accepted by M using any representations for the appearances of the transition function.

If the steps of the infinite game are played on general representations d of the transition function, then some subset τ' will be used in place of one of the defining sets in each step when OR chooses a strategy for the infinite game, call it S'. This may not be a strategy which could have been used by OR if the formulas d had been in IDNF.

By the second part of our lemma, however, there is always a defining set τ which is included in τ' which OR could have used if it had played the step on a representation which was in IDNF. Thus, there is some strategy S for OR in which the local strategy at each step is played on a representation of the lattice element which is in IDNF, and such plays of S are all present among the plays of S', We may think of the strategy tree $T_S(M,t)$ as being a "pruned" version of the strategy tree $T_{S'}(M,t)$. Thus, if all the plays of S' are winning plays, so are all the plays of S. We conclude that if the input tree t is accepted by M using any representations of the lattice elements $\delta(a_n, q_n)$, then it must also be accepted if these formulas are always in IDNF. As a result, we have the following theorem.

Theorem C1 (The Logical Equivalence Theorem). *In determining what language is defined by an alternating tree automaton M, it does not matter what representations are used for the lattice elements defined in connection with it.*

An immediate consequence of this theorem is the fact that we can freely construct a *dual automaton* $\tilde{M}$ from M if we interchange the roles of the two players OR and AND in the game $\Gamma(M,t)$. To do so we form $\tilde{M} = \langle Q, \Sigma, \tilde{\delta}, q_0, \tilde{\mathscr{F}} \rangle$, where the dual, $\tilde{\delta}$ is obtained from δ by interchanging $\vee$ and $\wedge$ and interchanging F and T. Here, $\tilde{\mathscr{F}}$ represents the complement of $\mathscr{F}$. We call this the "Duality Principle".

This principle immediately proves the Complementation Theorem. If OR fails to have a winning strategy for $\Gamma(M,t)$ then AND must have a winning strategy since Martin's Theorem assures that the game is determined. But $\tilde{M}$ is defined by interchanging $\wedge$ and $\vee$ in the transition function of M and complementing the acceptance condition. Playing the dual game $\Gamma(\tilde{M},t)$ in this form, we see that a winning strategy for AND for $\Gamma(M,t)$ directly becomes a winning first player strategy for the dual game. So for any input t, M fails to accept t if and only if $\tilde{M}$ does accept t.

The problem of determining the equivalence of two representations of lattice elements in $\mathscr{L}(X)$ is easily shown to be NP-complete. Thus, from the standpoint of complexity analysis, it is important when obtaining the transition function for a nondeterministic automaton N from that of an alternating automaton M, that the transition function for M does not have to be reduced to IDNF – it can be used directly to obtain the required state set as a consequence of the previous theorem.

Appendix D. The Regularity Theorem

Any nonempty regular language L of infinite k-ary trees must contain a regular tree t^*, that is, a tree in which there are only finitely many isomorphism classes of subtrees t_v^*, where t_v^* is the full subtree with root v. Let N be a nondeterministic automaton, say with subset acceptance, which accepts L. Project the alphabet Σ of L to the single letter alphabet Σ_1 and let N_1 be the corresponding automaton. A copy of N_1 can

make any possible transition of N. We can regard N_1 as an alternating automaton on the line exactly as in the "folding argument" for the emptiness problem. There is then a one-to-one correspondence between the k^n copies of N_1 at distance n from the origin and copies running on the k-ary tree.

Let N^* be the nondeterministic automaton on the line equivalent to N_1. The states of N^* are tuples of J-trees whose terminal nodes are labeled by sets of memories and the same memories are present on the terminal nodes of each J-tree in the given tuple A transition from one state to another is made by choosing a transition for each memory present and then updating the J-trees. A transition is thus specified by a transition-tuple $(\zeta_1, \tau_1, \ldots, \zeta_j, \tau_j)$ of pairs of memories and transitions. Let Δ be the complete transition graph of N^*. We think of the edges of Δ as being labeled by transition-tuples, where, of course, one edge may have several labels. Choose one label on an edge, erase the others, and let the resulting graph be Δ^*.

Since L is nonempty, N_1, and thus N^* accept. Since N^* is a nondeterministic automaton on the line, there is some finite loop γ in Δ^* such that the infinite path γ^* defined by taking a path γ_0 from the origin to the loop γ and then simply repeating γ, defines an accepting run S^* of N^*. In short, $\gamma^* = \gamma_0\gamma^\omega$.

For each pair (ζ, τ) consisting of a memory ζ and a transition τ which is possible on the last state in ζ, choose a letter a so that N can make the transition τ on a and think of this as a "stem", that is, a k-ary tree of height one with the root labeled by the pair (ζ, a) and the terminal nodes labeled by the memories present after making the transition τ.

We inductively construct a regular tree t^* associated with the path γ^* as follows. At the origin of Δ^*, the only memory present is the initial state q_0 of N. Take the stem associated with q_0 and the transition on the initial edge of γ^*. This is the tree t_1^*. Inductively, suppose that we have already constructed the finite tree t_n^* associated to the first n edges of γ^*. When we traverse the next edge, that edge associates a transition τ to each memory ζ present at a terminal node of t_n^*. Attach the stem associated with (ζ, τ) to each terminal node with memory ζ, thus placing the letter of the stem at that terminal node. This process defines an infinite tree t^*. Now replace each label (ζ, a) by simply the letter a. We keep t^* as the name of the resulting tree.

It is really clear that the infinite tree t^* is regular. In t^*, distinguish the levels $n_1, n_2 = n_1 + \ell$, $n_3 = n + 1 + 2\ell$, etc., which correspond to stages where we are at the first vertex in a repeat of the loop γ. If v is a vertex of t^* which is at a distinguished level or between two successive distinguished levels, associated to v the unique path α_v from a vertex u in the closest preceding distinguished level to the vertex v. Regard α_v as being labeled as a path in the k-ary tree. The *coordinate* of v is the pair (ζ, α_v) where ζ is the memory present at u. Any two points with the same coordinate have isomorphic subtrees in front of them by the form of the construction. There are only finitely many memories, each α_v is a path of length less than ℓ, and there are only finitely many vertices below the first distinguished level. Hence t^* is regular. The choice of stems defines an accepting run S of the original automaton N on t^* and thus $t^* \in L$.

Appendix E. Infinite automata

The purpose of this paper is to show that alternating automata are indeed really finite machines, that is, that they can be simulated by nondeterministic automata, and to investigate the complexity of this simulation. Our motivation for introducing the axiomatic framework for memory functions was to be able to consider cases where one could use a smaller amount of memory than the universally sufficient LAR. Once one views the arguments involved in the L-tree construction in terms of axioms however, one sees that everything works for "locally finite automata" without changing a word of the proof. It remains to be seen if there are any real applications in this direction, but we simply point out the generality of the L-tree argument in this appendix.

For focus, the reader might have in mind a case which does seem interesting – that of a pushdown automaton (PDA) reading an infinite input. Cohen and Gold [3] proved several results about pushdown automata on infinite words. The *total state* of a pushdown automaton is the pair (s, w) where s is the control state and w is the word written on the stack, so the set Q of possible total states of a PDA, in general is infinite. If we imagine the PDA reading an infinite input, then at any stage only finitely many states have occurred as the process develops. Of course, such a machine could be alternating.

In total generality, we may conceive of an alternating automaton working on $|K|$-ary trees as a tuple $\langle Q, \Sigma, \delta, q_0, \mathscr{F} \rangle$, where K is a countable set of directions, Q is a countable set of states, Σ is a countable input alphabet, $\delta : \Sigma \times Q \to \mathscr{L}(K \times Q)$ is an arbitrary transition function, and the acceptance condition $\mathscr{F}$ is any Borel subset of Q^ω. By Martin's Theorem, $\mathscr{F}$ being Borel ensures that $\Gamma(M, t)$ is determined. The definition ensures that δ is locally finite, that is, when reading an input letter in a given state, M has only finitely many choices and can send out only finitely many copies. We take this to be a characteristic of anything that can be thought of as a "machine".

The idea of a memory framework is exactly as before except we also drop the assumption that the sets involved are finite. Let $\mathscr{H}$ denote the set of all possible finite histories in the game $\Gamma(M, t)$, and let $\mu : \mathscr{H} \to Z$ be a function from $\mathscr{H}$ to another countable set Z. Now $\Omega = \{(G_i, R_i)\}_{i \in I}$ is allowed to have a countable index set. If I is infinite we suppose that $I = \mathbb{N}$.

In general, we let I^* denote the set of all *finite* sequences without repetition from I, and let the priority function be $\Pi : \mathbb{N} \times Z \to I^*$. In this case, the priority function of a pair (n, ζ) will be a list of the indices i such that $i \leqslant n$ and $\zeta \notin G_i$. We call $\langle \mu, \Omega, \Pi \rangle$ a memory framework and the statement of the axiom making $\langle \mu, \Omega, \Pi \rangle$ a stable memory framework for M is, word for word, exactly as before.

We construct the trees $L_{i,\lambda}$ simultaneously for all $i \in I$ by induction on the level n. The width of these trees now grows as the construction proceeds but they will always be locally finite. Since selection priorities are finite sequences, the history selection process poses no difficulty – at stage n consult the given sequence for (n, ζ) and select. The Branching Lemma remains valid. We also only require local finiteness for the

König's Lemma argument showing that S winning implies $\mathscr{C}(S)$ has good branching. What is interesting is that the Stability Axiom still is exactly what is needed to prove that $\mathscr{C}(S)$ has good branching implies that $\hat{S}$ is winning. Stability allows us to use the real force of a priority argument: For any particular losing play p and index i, the index i can only be injured finitely often. So we have proven the Strategy Uniformization Theorem in the infinite case: If OR has a winning strategy for $\Gamma(M,t)$ and $\langle \mu, \Omega, \Pi \rangle$ is a stable memory framework for M then OR has a μ-uniform winning strategy $\hat{S}$ which is a uniformization of S.

Acknowledgements

Thanks are due to Y. Gurevich, S. Zeitman and another referee for helpful comments on this paper.

References

[1] J.R. Büchi, On a decision method in restricted second-order arithmetic, in: *Proc. of the Internat. Congress an Logic, Methodology and Philosophy of Science*, 1960 (Stanford Univ. Press, Stanford, 1962) 1–12.

[2] J.R. Büchi, Using determinacy to eliminate quantifiers, Lecture Notes in Computer Science, Vol. 56 (Springer, Berlin, 1977) 367–378.

[3] R.S. Cohen and A.Y. Gold, Theory of ω-languages 1 – Characterizations of ω-context-free languages, *J. Comput. System Sci.* **15** (1977) 169–184.

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

[5] Y. Gurevich, Games people play, in: MacLane and Siefkes, eds., *The Collected Works of J. Richard Büchi* (Springer, Berlin 1990) 518–524.

[6] Y. Gurevich and L. Harrington, Trees, automata and games, in: *Proc. 14th ACM Symp. on Theory of Computing* (1982) 60–65.

[7] N. Klarlund, Constructions for tree automata, in: *Proc. of the IEEE Symp. on Logic in Computer Science* (1992) 382–393.

[8] R. McNaughton, Testing and generating infinite sequences by a finite automaton, *Inform. Control* **9** (1966) 521–530.

[9] R. McNaughton, Infinite games played in finite graphs, *Ann. Pure Appl. Logic* **65** (1993) 149–184.

[10] A.A. Muchnick, Games on infinite trees and automata with dead ends: A new proof of the decidability of the monadic theory of two successor functions, *Semiotics Inform.* **24** (1984) 17–24 (in Russian).

[11] D.E. Muller, Infinite sequences and finite machines, in: *Proc. 4th Ann. IEEE Symp. on Switching Circuit Theory and Logical Design* (1963) 3–16.

[12] D.E. Muller and P.E. Schupp, Alternating automata on infinite trees, *Theoret. Comput. Sci.* **54** (1987) 267–276.

[13] D.E. Muller, A. Saoudi and P.E. Schupp, Alternating automata, the weak monadic theory of trees and its complexity, *Theoret. Comput. Sci.* **97** (1992) 233–244.

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

[15] M.O. Rabin, Weakly definable relations and special automata, in: Y. Bar-Hillel, ed., *Mathematical Logic and Foundations of Set Theory* (North-Holland, Amsterdam, 1970) 1–70.

[16] M.O. Rabin, Automata on infinite objects and Church's problem, American Math. Soc., CBMS Lecture, Series, No. 13 (1972).

[17] S. Safra, On the complexity of ω-automata, in: *Proc. 29th IEEE Symp. on the Foundations of Computer Science* (1988) 319–327.
[18] S. Safra, Exponential determinization for ω-automata with strong fairness acceptance, in: *Proc. 24th ACM Symp. on the Theory of Computing* (1992) 275–282.
[19] R.I. Soare, *Recursively Enumerable Sets and Degrees* (Springer, Berlin, 1987).
[20] R.S. Streett, Propositional dynamic logic of looping and converse is elementary decidable, *Inform. Control* **54** (1982) 121–141.
[21] B.A. Trakhtenbrot and Ya. M. Barzdin', in: E. Shamir and L.H. Landweber, eds., *Finite Automata: Behavior and Synthesis*, Translated by D. Louvish (North–Holland, Amsterdam, 1973) 115–126.
[22] A. Yaknis and V. Yaknis, Extension of Gurevich–Harrington's restricted memory determinacy theorem: a criterion for the winning player and an explicit class of winning strategies, *Ann. Pure Appl. Logic* **48** (1990) 277–297.
[23] A. Yaknis and V. Yaknis, Gurevich–Harrington's games defined by finite automata, *Ann. Pure Appl. Logic* **62** (1993) 265–294.
[24] S. Zeitman, Unforgetful forgetful determinacy, *J. Logic Computation*, to appear.