The Rabin Index and Chain automata, with applications to automata and games

Sriram C. Krishnan *, Anuj Puri**, Robert. K. Brayton, Pravin P. Varaiya
Email: {krishnan,anuj,brayton,varaiya}@eecs.berkeley.edu

Department of EECS, University of California, Berkeley, CA-94720

Abstract. In this paper we relate the Rabin Index of an ω-language to the complexity of translation amongst automata, strategies for two-person regular games, and the complexity of controller-synthesis and verification of finite state systems, via a new construction to transform Rabin automata to Chain automata. The Rabin Index is the minimum number of pairs required to realize the language as a deterministic Rabin automaton (DRA), and is a measure of the inherent complexity of the ω-language. Chain automata are a special kind of Rabin automata where the sets comprising the acceptance condition form a chain. Our main construction translates a DRA with n states and h pairs to a deterministic chain automaton (DCA) with $n.h^k$ states, where k is the Rabin Index of the language. Using this construction, we can transform a DRA into a minimum-pair DRA or a minimum-pair deterministic Streett automaton (DSA), each with $n.h^k$ states. Using a simple correspondence between tree automata (TA) and games, we extend the constructions to translate between nondeterministic Rabin and Streett TA while simultaneously reducing the number of pairs; for the class of "trim" deterministic Rabin TA our construction gives a minimum-index deterministic Chain TA, or a minimum-pair DRTA or DSTA, each with $n.h^k$ states, where k is RI of the tree-language.
Using these results, we obtain upper bounds on the memory required to implement strategies in infinite games. In particular, the amount of memory required in a game presented as a DRA, or DSA, is bounded by nh^k, where k is the RI of the game language.

1 Introduction

Deterministic ω-automata have varying levels of complexity depending on the acceptance condition. A set of states is an accepting set if it satisfies the acceptance condition. Deterministic Buchi Automata are the simplest ω-automata. They have the special property that every set which contains an accepting set is also accepting. For deterministic ω-automata with more general acceptance condition (such as Rabin or Streett acceptance condition), this is no longer true. In general, a set may be accepting, a set containing that not accepting, but a set containing that in turn accepting, and so on. The number of alternations between accepting and non-accepting sets is an indication of the inherent complexity in the ω-automata, and has a firm topological basis [21, 9, 11, 2]. It is also closely related to the Rabin

* supported by NSF/DARPA Grant MIP-8719546
** supported by the California PATH program and NSF under grant ECS9417370

Index—the minimum number of pairs required for a deterministic Rabin automata (DRA) to accept the same language. Chain automata (a syntactic variant of Parity automata [5, 6]) are of fundamental interest because they capture the alternation in the structure of their acceptance condition. A deterministic Chain automaton can be trivially complemented, and admits a straightforward minimization in the number of pairs to the Rabin Index of the language it accepts. The chain acceptance condition is also significant because μ-calculus model checking is polynomially equivalent to chain TA-emptiness [6].

Our main construction converts a DRA with n states and h pairs, into a deterministic Chain automaton with nh^k states and k pairs where k is the Rabin Index of the language. Using this construction, it becomes possible to transform a deterministic Rabin or Streett ω-automaton with n states and h pairs into a minimum-pair deterministic Rabin or Streett automaton with nh^k states. Our construction using Chain automata also simplifies the construction in [11]. By employing a simple correspondence between tree-automata and ω-automata, we show that a "trim" deterministic Rabin tree automaton or Streett tree automaton can be converted to a minimum-pair deterministic chain tree automaton (DCTA), DRTA, or DSTA with the same complexity as before.

Chain automata are also of special significance in infinite two-person games. A game is played by two players on an ω-automaton (also called the Game Automaton) with a certain game language. In general, due to the infinite nature of the game, the strategies must be functions of the states that have been visited in the past. But for chain automata, both players have strategies that are functions of only the current state of the game (also called memory-less strategies) [5, 14, 20]. Using our construction, it is possible to get upper bounds on the amount of memory needed by the players to implement their strategies. In particular, in a game represented by a DRA or DSA, the winning player has a strategy which requires memory of size at most nh^k, where k is Rabin Index of the game language. Since there is a close relationship between strategies and trees, our results also imply bounds on the "size" of regular trees accepted by tree automata.

Our constructions have applications to the synthesis and verification of compositional systems. In practice, hardware designs and protocols are conceived of as a system of interacting components. The Rabin Index of such compositional systems, the alternation complexity of their ω-language, is often much lower than the number of pairs they are syntactically specified with. Our constructions can be used to convert automata to the minimum-pair automata, providing a computational advantage when used in conjunction with algorithms that are exponential in the number of pairs. For example, deciding the controller synthesis problem is exponential in the number of pairs [4, 15]. Our results make the algorithm a function of the Rabin Index—of practical importance when the Rabin Index is smaller than the number of pairs. Our pair minimization algorithms also have applications in the context of compositional verification.

There have been two previously published algorithms to convert DRA to DCA. For a DRA with n states and h pairs, Carton's [1] construction gives a DCA with $O(n^h 4^{2h^2})$ states. Emerson and Jutla [5] observe that an algorithm of Safra [18] to convert DRA to DSA can be modified to yield a DCA with $n.h^h$ states.

Section 2 presents our notation and definitions. In Section 3, we present our main contribution: the transformation from a DRA to a minimum-index DCA. In Section

4 we show how to convert a DRA or DSA into a minimum-pair DRA or DSA. In Section 5, we extend the constructions to TA by employing a correspondence between tree automata and games. In Section 6, we discuss the relevance of our results to two-person games. In Section 7, we discuss the practical significance of the Rabin Index to the complexity of synthesis and verification.

2 Preliminaries

2.1 ω-automata

An ω-automaton [19] over a finite alphabet, Σ, is $\mathcal{A} = \langle T, \phi \rangle$, where T is a transition structure and ϕ is the acceptance condition. The transition structure is $T = \langle Q, q_0, \Sigma, \delta \rangle$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, and $\delta : Q \times \Sigma \to Q$ is the transition function. For most of this paper we will be concerned with complete deterministic ω-automata (DOA), i.e. δ is a *total* function. The state transition graph (STG) is the directed graph on the vertex set Q denoted by the transition structure T. The transition function δ defines the Σ-labeled edges.

Definition 1. A word $\sigma \in \Sigma^\omega$ has the run $r_\sigma \in Q^\omega$, where $r_\sigma(0) = q_0$ and $\delta(r_\sigma(i), \sigma(i)) = r_\sigma(i+1)$.

The infinity set of the run r_σ, denoted $inf(r_\sigma)$, is the set of states that are visited infinitely often in r_σ. The acceptance condition ϕ is a Boolean formula, where the Boolean variables are the states $Q = \{q_1, \ldots, q_n\}$.

Definition 2. A Boolean formula is generated by the following rules
1) $q_i \in Q$ is a Boolean formula
2) If ϕ_1, ϕ_2 are Boolean formulae, then $\neg\phi_1$, $\phi_1 \vee \phi_2$,and $\phi_1 \wedge \phi_2$ are Boolean formulae.

The truth of $q_i \in Q$ is determined by the run r_σ. For $C \subseteq Q$ define the assignment $q_i = true$ provided $q_i \in C$. Let $\phi[C]$ denote the truth value of ϕ under this assignment.

Definition 3. The language generated by the ω-automaton $\mathcal{A} = \langle T, \phi \rangle$, denoted $\mathcal{L}(A)$, is $\{\sigma | \sigma \in \Sigma^\omega$ and $\phi[inf(r_\sigma)] = true\}$.

Definition 4. We define various types of Boolean formulae which are used to define acceptance criteria :
1) A disjunctive formula (DF) is a disjunction of Boolean variables, i.e., $\phi = q_{i_1} \vee \ldots \vee q_{i_k}$.
2) A Rabin formula is $\phi = \vee_{i=1}^n (L_i \wedge \neg(\overline{U_i}))$ where $L_i, U_i, 1 \leq i \leq n$ are DF.
3) A Streett formula is $\phi = \wedge_{i=1}^n (L_i \vee \neg(\overline{U_i}))$ where $L_i, U_i, 1 \leq i \leq n$ are DF.

A Buchi automaton (DBA) is $A = \langle T, \phi \rangle$, where ϕ is a disjunctive formula. A Rabin automaton (DRA) is defined by a Rabin formula and a Streett automaton (DSA) with a Streett formula. Since each DF identifies a set of states, the Buchi condition is often called the final states. The Rabin and Streett acceptance conditions are also referred to as a set of **pairs** (L_i, U_i) of subsets of states. Informally, a run r_σ in a DRA is accepting if for some pair, $inf(r_\sigma)$ "touches" L_i and is contained in U_i. A

DRA may be complemented to yield a DSA (and vice-versa) on the same transition structure with the same number of pairs, by replacing each pair (L_i, U_i) by $(\overline{U_i}, \overline{L_i})$. Thus DSA and DRA are syntactic complements of each other.

Remark. Our syntax for the Rabin and Streett condition differs from the standard definition in the literature, where a Rabin formula is $\vee_{i=1}^{n}(L_i \wedge \neg U_i)$; a run is accepting (for Rabin acceptance), if for some i, it visits L_i (the GREEN states) infinitely often, but U_i (the RED states) only finitely often. Complementing U_i translates between the standard syntax and ours.

2.2 Rabin Index and Chain automata

Definition 5. The chain acceptance condition [20] is represented by a formula $\phi = \vee_{i=1}^{n}(F_i \wedge \neg E_i)$ where $F_i, E_i, 1 \leq i \leq n$ are DF, and $E_1 \subset F_1 \subset E_2 \subset F_2 \subset \ldots \subset E_n \subset F_n$.

A run in a chain automaton is accepting if for some i, it visits E_i finitely often and F_i infinitely often. In the traditional syntax of the Rabin condition, the chain condition is a particular case where the sets form a chain under set inclusion.

For an ω-automaton $A = \langle T, \phi \rangle$, define $\mathcal{C}$ to be the set of Strongly Connected Sets[3] (SCSs) of T. There is a partial order on $\mathcal{C}$ induced by set inclusion ($\subset$). A strongly connected set S is **accepting** provided $\phi[S] = True$, and **rejecting** otherwise. The polarity of a SCS S is "+" if S is accepting, and "-" if S is rejecting.

A **chain** is a linearly ordered set $S_1 \subset S_2 \subset \cdots \subset S_m$, where each S_i is a SCS. In an alternating chain, the polarity of S_{i+1} is opposite of S_i. The **index** of an alternating chain of length m is $\lceil \frac{m}{2} \rceil$. A positive (negative) chain is an alternating chain in which the polarity of the first set in the chain S_1 is "+"("-"). The **Rabin Index** (RI) of a Deterministic ω-automaton is the index of the longest positive chain. The **Streett Index** (SI) is the index of the longest negative chain. The difference between the SI and RI (and vice-versa) is at most one.

Wagner [21] and Kaminski [9] showed that the RI (SI) is also the minimum number of pairs in a deterministic Rabin (Streett) automaton required to realize the same language.

Figure 1 shows the SCSs of an ω-automaton. An edge from i to j indicates that $i \subset j$. The longest negative chain is $(1, 2, 5, 6)$, and the longest positive chain is $(4, 5, 6)$. Therefore, the RI and the SI are both 2.

For an ω-automaton A, we define the **superset closed** SCSs to be $sup(A) = \{c |$ the polarity of c is "+", and if $c \subset d$, then the polarity of d is "+"$\}$. In Figure 1, $sup(A) = \{6, 7, 8\}$.

The superset closed sublanguage is $supLan(A) = \{\sigma | \; inf(r_\sigma) \in sup(A)\}$.

Given an ω-automaton $\mathcal{A} = \langle T = (Q, q_0, \Sigma, \delta), \phi \rangle$, a state $q \in Q$ is termed **final** provided every SCS C containing q is accepting.

The terminology, final state, comes from BA where every SCS containing a final state is accepting; every accepting SCS is superset closed. Landweber [19] showed that the language of a DOA can be equivalently realized as a DBA if and only if every accepting SCS is superset closed. Therefore the class of languages accepted by DBA form a strict subset of the class of ω-regular languages, and are contained in

[3] $C \subseteq Q$ is a SCS if there is a path in the STG between any two states of C

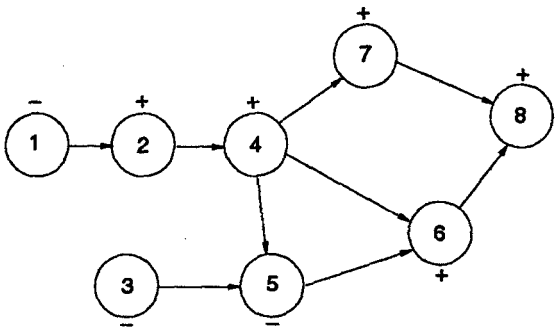

Fig. 1. Hasse Diagram of SCSs of an automaton

the Borel class G_δ [19] (the ω-regular languages are the class of languages accepted by DRA or DSA).

2.3 Tree automata and Games

We consider finite automata on labeled, infinite binary trees[4]. The set $\{0,1\}^*$ can be viewed as the infinite binary tree, where the root node is the empty string ϵ and each node η has two successors: the 0-successor $\eta 0$ and the 1-successor $\eta 1$ [4]. An infinite path through the tree is a sequence $\eta \in \{0,1\}^\omega$. If Σ is a finite alphabet, a Σ-valued tree is a labeling $t : \{0,1\}^* \to \Sigma$.

A finite automaton $\mathcal{A}$ on infinite binary Σ-valued trees (henceforth tree automaton) is $\mathcal{A} = \langle T, \phi \rangle$, where T is a transition structure and ϕ is the acceptance condition. The transition structure is $T = \langle Q, q_0, \Sigma, \delta \rangle$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, and $\delta : Q \times \Sigma \to Q \times Q$ is the transition function. The acceptance condition is a Boolean formula over the states Q (as for ω-automata).

A *run* of $\mathcal{A}$ on a Σ-valued tree t is a Q-valued tree $r_t : \{0,1\}^* \to Q$, such that $r_t(\epsilon) = q_0$ and for all $\eta \in \{0,1\}^*$, $\delta(r_t(\eta), t(\eta)) = (r_t(\eta 0), r_t(\eta 1))$. We say that $\mathcal{A}$ accepts tree t provided in the run r_t of t on $\mathcal{A}$, for every infinite Q-labeled path α of r_t, $\phi[inf(\alpha)] = true$.

Using Rabin acceptance condition leads to Rabin Tree automata (RTA), and Streett acceptance condition to Streett Tree automaton (STA).

Definition 6. We say a TA $\mathcal{A} = \langle T = (Q, q_0, \Sigma, \delta), \phi \rangle$ is **trim** provided for every $s \in Q$ the TA $\mathcal{A}_s = \langle T = (Q, s, \Sigma, \delta), \phi \rangle$ accepts some tree.

There is a close relation between tree automata and two-person Gale-Stewart games.

Definition 7. Given finite alphabets A and B, a regular two-person *Gale-Stewart* game is represented by an ω-regular language $\Gamma \subseteq (A \times B)^\omega$ (the game language) [19]. A single *play* of the game is as follows: player I picks some $a_0 \in A$, then II picks some $b_0 \in B$, followed by I picking some $a_1 \in A$, and so on in turns. Player I wins if the resulting ω-word $(a_0, b_0)(a_1, b_1) \ldots$ is in Γ; otherwise II wins. A strategy for I is a function $f : B^* \to A$, indicating to I what to play next. A strategy for

[4] extension to k-ary trees is straightforward

II is similarly a function $g : A^+ \to B$. We say that f is a winning strategy for I if every play that results from I playing according to f is a win for I.

We call an ω-automaton representing a game language Γ as above, a **Game Automaton**. A tree-automaton can be viewed as a game between two players: the Σ-player and the path-player (or $\{0,1\}$-player). More formally, given a TA $\mathcal{A} = \langle (Q, q_0, \Sigma, \delta), \phi \rangle$, define the corresponding game automaton as $\mathcal{A}' = \langle (Q, q_0, \Sigma', \delta'), \phi \rangle$ where $\Sigma' = \Sigma \times \{0,1\}$, and $\delta(q, \sigma) = (\delta'(q, (\sigma, 0)), \delta'(q, (\sigma, 1)))$. Similarly, given a deterministic game automaton on alphabet $\Sigma \times \{0,1\}$, a corresponding DTA can be inferred. Hence for a DTA the corresponding Game automaton (GA) is well-defined and unique and vice-versa.

A Σ-valued tree accepted by a tree automaton can be viewed as a strategy for the Σ-player in the corresponding game automaton. Hence, Player I has a winning strategy in the game automaton iff the corresponding tree automaton has a non-empty language. Emerson [3] showed that every RTA with non-empty language accepts a tree that can be "embedded" in the automaton. It follows that in such an automaton the Σ-player has a strategy that is just a function from $Q \to \Sigma$, termed a memory-less strategy by McNaughton [14].

3 Converting DRA to Deterministic Chain automata

We first review some constructions from [11] to isolate the superset closed sublanguage of Rabin automata and Streett automata.

3.1 Isolating the superset closed sublanguage

Our basic constructions achieve the following: Given a deterministic automaton $\mathcal{A} = \langle T, \phi \rangle$, we construct a deterministic automaton $\mathcal{A}' = \langle T', \phi' \rangle$ and identify a subset of states $S \subseteq Q'$ with the following properties: 1) $L(\mathcal{A}') = L(\mathcal{A}) \setminus supLan(\mathcal{A})$, and 2) $L(\langle T', S \rangle) = supLan(\mathcal{A})$ where $\langle T', S \rangle$ denotes the DBA on transition structure T' with final states S. We term this construction the exclusion construction; *RabExcl* is the exclusion construction for Rabin automata and *StrExcl* for Streett automata.

Rabin automata

We defined the notion of a final state in Section 2. We showed in [10] how to determine in polynomial time if a state in a RA is final.

Lemma 8 [11]. *Given a DRA $\mathcal{A}$, a SCS C is superset closed if and only if $C \cap F \neq \emptyset$, where F is the set of final states of the DRA.*

RABIN EXCLUSION CONSTRUCTION (*RabExcl*)
INPUT: DRA $\mathcal{A} = \langle T = (Q, q_0, \Sigma, \delta), \phi \rangle$ where ϕ consists of h pairs (L_i, U_i).
OUTPUT: DRA $\mathcal{A}' = \langle T, \phi' \rangle$ where ϕ' consists of h pairs, $(L_i', U_i') = (L_i \setminus F, U_i \setminus F)$, and F is the set of final states of $\mathcal{A}$.

Lemma 9 [11]. RabExcl *has the property that $L(\mathcal{A}') = L(\mathcal{A}) \setminus supLan(\mathcal{A})$, and $L(\langle T, F \rangle) = supLan(\mathcal{A})$, where $\langle T, F \rangle$ is a DBA with final states F.*

Note that the superset closed sublanguage of a DRA can be isolated by identifying final states on the *same* transition structure as the DRA.

Streett Automata

Consider a DSA $\mathcal{A} = \langle T, \phi \rangle$ with h pairs. Without loss of generality we can assume $L_i \cap U_i = \emptyset$. Let $\mathcal{A}_i = \langle T, \phi_i \rangle$ denote the DSA with a single pair: the i^{th} pair, (L_i, U_i), of ϕ.

Lemma 10 [11]. *For DSA $\mathcal{A}_i$, a SCS C is superset closed if and only if $C \cap F_i \neq \emptyset$, where F_i is the set of final states of $\mathcal{A}_i$.*

Theorem 11 [11]. *For a DSA $\mathcal{A}$, a SCS C is superset closed if and only if $C \cap F_i \neq \emptyset$ for each i, where F_i is the set of final states of $\mathcal{A}_i$.*

The Streett exclusion construction is given below. The basic idea is to make h copies of the transition structure and direct the transitions from the states in F_i in the i^{th} copy to the corresponding states in the $(i+1)^{st}$ copy.

STREETT EXCLUSION CONSTRUCTION (*StrExcl*)

INPUT: DSA $\mathcal{A} = \langle T = (Q, q_0, \Sigma, \delta), \phi \rangle$ where ϕ consists of h pairs (L_i, U_i).

OUTPUT: DSA $\mathcal{A}' = \langle T' = (Q', q_0', \Sigma, \delta'), \phi' \rangle$ and $S \subseteq Q'$

Let F_i denote the final states for the DSA $\mathcal{A}_i$

$Q' = Q \times \{1, \ldots, h\}$, $q_0' = (q_0, 1)$

$$\delta'((q,j),a) = \begin{cases} (\delta(q,a), j) & \text{if } q \notin F_j \\ (\delta(q,a), j+1) & \text{if } q \in F_j \text{ and } j < h \\ (\delta(q,a), 1) & \text{if } q \in F_h \end{cases}$$

$S = \{(q,i) | q \in F_i \text{ and } 1 \leq i \leq h\}$, ϕ' consists of $h+1$ pairs (L_i', U_i'), where

$L_i' = \{(q,j) | q \in L_i \text{ and } 1 \leq j \leq h\}$,

$U_i' = \{(q,j) | q \in U_i \text{ and } 1 \leq j \leq h\}$, and $(L_{h+1}', U_{h+1}') = (\emptyset, \overline{S})$.

In contrast to the Rabin case, for DSA we have to expand the transition structure and add an extra pair $(\emptyset, \overline{S})$ to exclude the superset closed language.

From Theorem 11 it follows that:

Lemma 12. *StrExcl has the property that $L(\mathcal{A}') = L(\mathcal{A}) \setminus supLan(\mathcal{A})$ and $supLan(\mathcal{A}) = L(\langle T', S \rangle)$. $\mathcal{A}'$ has $n.h$ states and $h+1$ pairs, where n and h are respectively the number of states and pairs in $\mathcal{A}$.*

From the properties of *RabExcl* and *StrExcl* it follows that:

Lemma 13. *Given a DSA (DRA) $\mathcal{A} = \langle T, \phi \rangle$, and $\mathcal{A}' = \langle T', \phi' \rangle$ obtained by applying* StrExcl *(*RabExcl*) on $\mathcal{A}$, $supLan(\mathcal{A}) \subseteq supLan(\overline{\mathcal{A}'})$.*

3.2 Computing the minimum-chain-length DCA

Using our constructions to isolate the superset closed sublanguage, we devise an algorithm to transform DRA to minimum-chain-length DCA.

Given a DRA we successively apply complement, *StrExcl*, complement, and *RabExcl* until the language of the automaton is empty. In each iteration we reduce the index of each positive chain by one. The number of iterations is bounded by the index of the longest positive chain—the RI of the language.

StrExcl and *RabExcl* may be seen as reversing the polarity of the SCSs that are superset closed ("peels" them off). Therefore any positive chain ending in a superset closed SCS has index one lower after applying the exclusion construction. The outer SCSs, in alternating chains ending in rejecting SCSs, are peeled off in complemented form, i.e. after complementing the automaton.

More formally, the construction is:

INPUT: DRA $\mathcal{A} = \langle T, \phi \rangle$, where ϕ consists of h pairs
$i = 0$; $\mathcal{A}_0 = \mathcal{A}$
Repeat
  i = i + 1;
  1. $\mathcal{A}_{i1} = \overline{\mathcal{A}_{i-1}}$ "Complement"
  2. $(\mathcal{A}_{i2}, E_i) = StrExcl(\mathcal{A}_{i1})$ "Exclude *supLan*; expands Tr. Str."
  3. $\mathcal{A}_{i3} = \overline{\mathcal{A}_{i2}}$ "Complement"
  4. $(\mathcal{A}_i, F_i) = RabExcl(\mathcal{A}_{i3})$ "Exclude *supLan*"
until $L(\mathcal{A}_i) = \emptyset$
Return DCA $\langle (Q_k, q_{0_k}, \Sigma, \delta_k), \psi \rangle$, where ψ in chain form is
$\pi_1^{-1}(E_1) \subset \pi_1^{-1}(F_1) \subset \ldots \subset \pi_i^{-1}(E_i) \subset \pi_i^{-1}(F_i) \subset \ldots \subset \pi_k^{-1}(E_k) \subset \pi_k^{-1}(F_k)$, where k is the RI of the language.

For example, in the automaton whose SCSs are represented in Figure 1, during the first iteration: in step 2 we pick up no SCSs, in step 4 we pick up 6,7, and 8. During the second iteration, in step 2 we pick up SCSs 3 and 5, and in step 4 we pick up 2 and 4. At this stage all the SCSs are of negative polarity and the language is empty.

Each iteration involves a complement into a DSA (on the same transition structure), *StrExcl* (expands the transition structure h-fold), complement into a DRA (on the same transition structure), and *RabExcl* (on the same transition structure).

Notation: The automaton $\mathcal{A} = \langle T = (Q, q_0, \Sigma, \delta), \phi \rangle$. The automaton obtained after the i^{th} iteration is $\mathcal{A}_i = \langle T_i = (Q_i, q_{0_i}, \Sigma, \delta_i), \phi_i \rangle$ where (it follows from Lemma 14 that) $Q_i = Q_{i-1} \times \{1, \ldots, h\}$; after the k^{th} iteration $Q_k = Q \times \{1, \ldots, h\}^k$ and the map $\pi_i : Q_k \to Q_i$ is the natural projection map.

Before we prove the correctness of the construction we observe two preliminary lemmas.

Lemma 14. *In each iteration of the construction, the pair added in step 2 is redundant after step 4.*

Proof. *StrExcl* excluded the superset closed sublanguage at step 2, adding a pair $(\emptyset, \overline{E_i})$. On complementing, this becomes a Rabin pair (E_i, Q_i). Lemma 13, implies that the state set returned at step 4, F_i, contains E_i, i.e. $F_i \supseteq E_i$. In *RabExcl* F_i is deleted from each set in each pair, transforming the added Rabin pair (E_i, Q_i) to $(\emptyset, Q_i \setminus F_i)$—a pair with empty language that can be deleted. □

Hence at the end of each iteration we have exactly h pairs in $\mathcal{A}_i$.

Lemma 15. *Given $\sigma \in \Sigma^\omega$, $0 \leq i \leq k$, and $q \in Q_i$, $q \in inf(r_i)$ if and only if for some $s \in Q_k$, $s \in inf(r_k)$ and $\pi_i(s) = q$, where r_j is the run of σ on $\mathcal{A}_j$.*

Proof Sketch. By induction on $k - i$. Observe that *StrExcl* "copies" the transition structure. □

Theorem 16. *Given a DRA $\mathcal{A} = \langle T = (Q, q_0, \Sigma, \delta), \phi \rangle$ where ϕ has h pairs, the construction returns an equivalent DCA with $n.h^k$ states and chain length $2k$ (i.e., k pairs), where k is the RI of the language.*

Proof sketch. Observe the following:

1. Lemma 13 implies that $\pi_1^{-1}(E_1) \subset \pi_1^{-1}(F_1) \subset \pi_2^{-1}(E_2) \subset \pi_2^{-1}(F_2) \subset \ldots \subset \pi_k^{-1}(E_k) \subset \pi_k^{-1}(F_k)$, i.e. form a chain.
2. Lemma 14 implies that after k iterations in $\mathcal{A}_k$, $|Q_k| = |Q|.h^k = n.h^k$.

From Lemmas 9, 12, and 13 it follows that:

Lemma 17. *For each i, $1 \leq i \leq k$, $L(\mathcal{A}_{i-1}) = L(\mathcal{A}_i) \cup L(\langle T_i, (\neg E_i \wedge F_i) \rangle)$, where k is the RI of $L(\mathcal{A})$.*

Therefore from Lemma 17 and Lemma 15 we have for each i, $1 \leq i \leq k$,

$$L(\mathcal{A}) = L(\langle T_i,\ \phi_i \vee \bigvee_{j=1}^{i} (\neg \pi_j^{-1}(E_j) \wedge \pi_j^{-1}(F_j)) \rangle) \tag{1}$$

Lemma 18. *For each i, $0 \leq i \leq k-1$, the Rabin Index of $\mathcal{A}_i$ is one less than the Rabin Index $\mathcal{A}_{i-1}$.*

Proof sketch. Consider any positive chain in $\mathcal{A}_{i-1}$. It either ends in an accepting ("+") SCS or a rejecting ("−") SCS. If it ends in a "+" SCS its polarity is inverted in step 4, reducing the index of the chain by 1.

Let the chain that ends in a rejecting SCS be: $\ldots + - + -$. In step 1 the chain reverses in polarity. After step 2 it is: $\ldots - + - -$. After step 3 the chain again reverses in polarity. After step 4 the chain is: $\ldots + - - -$, and has index 1 lower. □

Since each iteration reduces the index of each positive chain by 1, we have k—the RI of the language—iterations, before $L(\mathcal{A}_k) = \emptyset$; at this stage ϕ_k is unsatisfiable and can be deleted.

Thus we get $L(\mathcal{A}) = L(\langle T_k, \bigvee_{j=1}^{k} (\neg \pi_j^{-1}(E_j) \wedge \pi_j^{-1}(F_j)) \rangle)$. The acceptance condition in chain form is:

$$\pi_1^{-1}(E_1) \subset \pi_1^{-1}(F_1) \subset \ldots \subset \pi_k^{-1}(E_k) \subset \pi_k^{-1}(F_k) \tag{2}$$

□

Our transformation from DRA to DCA has complexity $DR(n,h) \rightarrow DC(n.h^k, k)$. Carton's [1] transformation gives an equivalent DCA with $n^h 4^{2h^2}$ states. Emerson and Jutla [5] suggest an adaptation of a construction of Safra from [18] to convert a DRA to DSA, to transform DRA to DCA; they achieve a transformation $DR(n,h) \rightarrow DRC(n.h^h, h)$. Our transformation is better when $k < h$ and matches theirs in the worst case.

Also, our construction is optimum in the sense that we cannot hope to get a transformation that is polynomial in k because:

1. There is an exponential lower bound on translating DSA to DRA [17]. We show in the next section that we can obtain a transformation from DSA to DRA of the same complexity as for DRA to DCA.
2. We showed in [11] that it is NP-hard to determine the RI of a language specified as a DRA or DSA.

4 Translating between different automata

In this section we use the construction of Section 3 to translate a DRA or DSA into a minimum-pair DRA or DSA. We do this by exploiting the ease of complementation and chain-length minimization of DCA; and straightforward translation of DCA into DRA.

DCA can be trivially transformed to DRA: each Chain pair (E_i, F_i) gets transformed to a Rabin pair $(F_i, \overline{E_i})$.

Also, DCA can be trivially complemented. Given a DCA with h pairs (E_i, F_i), with chain $E_1 \subset F_1 \subset E_2 \subset F_2 \subset \ldots \subset E_h \subset F_h$, the complementary DCA has $h+1$ pairs $(\emptyset, E_1), (F_1, E_2), \ldots, (F_{h-1}, E_h), (F_h, Q)$, where Q is the state set of the automaton.

Furthermore, the chain length in a DCA can be easily minimized to yield an equivalent DCA on the same transition structure with chain length twice RI of the language. We briefly sketch the basic idea behind a polynomial time algorithm below.

We first define the parity acceptance condition [5, 6]. It consists of disjoint sets $M_1, N_1, M_2, N_2, \ldots, M_h, N_h$. A run is accepting provided for some i, $1 \leq i \leq h$, it visits N_i infinitely often but not any set to the left of it. More precisely, run r is accepting provided for some i, $inf(r) \cap N_i \neq \emptyset$ and $inf(r) \cap ((\bigcup_{j=1}^{i} M_j) \cup (\bigcup_{j=1}^{i-1} N_j)) = \emptyset$. The chain acceptance condition can be readily translated to the parity acceptance condition: for $1 \leq i \leq h$, $N_i = F_i \setminus E_i$, $M_1 = E_1$, and for every i, $2 \leq i \leq h$, $M_i = E_i \setminus F_{i-1}$. Similarly given the parity sets, the corresponding chain sets are formed by taking increasing unions starting from the left.

To minimize the chain in a chain automaton, we first convert it to parity acceptance. The idea then is to "grow" positive chains starting from states in N_h. We start with N_h and the subgraph induced by $Q \setminus E_h$, and the find the SCCs[5] of the states in N_h. We gradually let in sets to the left in the parity acceptance condition and recompute SCCs. At each step we have a set of chains. Depending on the nature of the SCCs of the new states let in, we either delete these states, add them to existing chains/start a new chain, possibly promoting the state to the right in the parity sets. When the procedure finishes we have exactly twice the RI number of sets, and a positive chain attesting to the RI of the language (see [1] for more details).

Using the transformation from DRA to minimum-chain-length DCA and the properties of DCA outlined above we have a means to convert a DRA to either a minimum-pair DRA or DSA, as shown in Figure 2. Note that the number of pairs in the DCA obtained on complementing a minimum-pair DCA can be at most 2 more than its minimum.

[5] an SCC is a maximal SCS

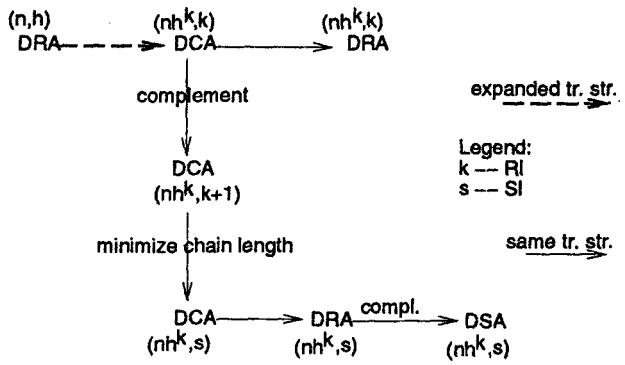

Fig. 2. Translation map

While the constructions presented in the previous section share the same spirit as those in [11], using chain automata provides for a more elegant and uniform treatment of the conversion between different automata.

5 Translating Tree automata

In this section we extend the applicability of the constructions of the previous two sections to nondeterministic TA.

Lemma 19. *Given a nondeterministic TA* $\mathcal{A} = \langle T = (Q, q_0, \Sigma, \delta), \phi \rangle$, *there exists a DTA* $\mathcal{A}' = \langle T' = (Q, q_0, \Sigma', \delta'), \phi \rangle$, *and a projection* $\pi : \Sigma' \to \Sigma$ *such that* $L(\mathcal{A}) = \pi(L(\mathcal{A}'))$.

Proof. The construction is simple and well known. $\Sigma' = \Sigma \times Q \times Q$. The projection $\pi(a, q_1, q_2) = a$. $\delta'(q, (a, q_1, q_2)) = (q_1, q_2)$ provided $(q_1, q_2) \in \delta(q, a)$. $\mathcal{A}'$ can be completed by the addition of a "dead" state. □

Lemma 20. *Suppose* $\mathcal{A}$ *and* $\mathcal{B}$ *are DTA, and* $\mathcal{A}_G$ *and* $\mathcal{B}_G$ *are the corresponding Game automata. If* $L(\mathcal{A}_G) = L(\mathcal{B}_G)$ *then* $L(\mathcal{A}) = L(\mathcal{B})$.

Proof. Suppose $L(\mathcal{A}_G) = L(\mathcal{B}_G)$, and $t \in L(\mathcal{A})$. We need to show $t \in L(\mathcal{B})$. Since $\mathcal{A}$ ($\mathcal{B}$) is deterministic and complete, t has a unique run r_t (r'_t) in $\mathcal{A}$ ($\mathcal{B}$). Let π be any infinite path starting from ϵ in $\{0,1\}^*$. Since $\mathcal{A}$ is deterministic the path in the run tree r_t corresponding to path π denotes a unique sequence σ in $(\Sigma \times \{0,1\})^\omega$ such that $\sigma \in L(\mathcal{A}_G)$. Since $L(\mathcal{B}_G) = L(\mathcal{A}_G)$, the path in the run r'_t corresponding to path π satisfies the acceptance condition. Therefore $t \in L(\mathcal{B})$. □

The above Lemma asserts that the Game Automaton can be used to translate between DTA.

Lemmas 20 and 19 imply that we can use the constructions of the previous section to convert RTA to CTA and hence to either a STA or RTA, while simulatneously reducing the number of pairs. With TA, the RI upon transforming to the corresponding GA could in general be greater than the actual RI of the tree language.

However for a trim DRTA, minimizing the number of pairs in the GA also yields a tree automaton with the minimum number of pairs.

Theorem 21. *Given a trim DRTA $\mathcal{A}$, let $\mathcal{A}_G$ be the corresponding GA. Suppose $\mathcal{B}_G$ is obtained from $\mathcal{A}_G$ by minimizing the number of pairs. Then the DRTA $\mathcal{B}$ corresponding to $\mathcal{B}_G$ has the minimum number of pairs.*

Proof sketch. It will be sufficient to show that any equivalent DRTA $\mathcal{M}$ has at least as many pairs as in the DRA $\mathcal{B}_G$. $\mathcal{M}$ can be trimmed giving an equivalent automaton with the same number of pairs.

We claim that $L(\mathcal{B}_G) = L(\mathcal{M}_G)$. Observe that in a trim TA every sequential accepting run is "part" of some accepting tree run. In a DTA every sequential run is a unique sequence in $\Sigma \times \{0,1\}$. Since the GA is exactly the set of accepting sequential runs it follows that $L(\mathcal{B}_G) = L(\mathcal{M}_G)$. □

6 Strategies in Games

Two-person games are interesting from an applications viewpoint because the synthesis-problem [16] can be seen as a game between the controller and the disturbance [20]. Strategies that are easier to implement are desirable. A particularly simple form of a strategy is a memory-less strategy where the strategy is a function of just the current state set of the game automaton [13].

The existence of a winning strategy for one player being implied by the non-existence of a winning strategy for the other player is a non-trivial and non-obvious fact that holds in particular of all two-person perfect information games. In a game presented as a Rabin automaton, if player I has a winning strategy, then I also has a memory-less winning strategy [3]. If player II has a winning strategy, the question is how complex is the strategy function. Gurevich and Harrington [7] showed that it is sufficient to remember the *latest appearance record* (LAR) of the states visited and not the entire sequence of states. This bounds the memory required to implement the strategy by $|Q!|$. Depending on the acceptance condition used, the strategy can be implemented with less memory than the complete LAR.

The chain acceptance condition has the special property that the union of two rejecting (accepting) SCSs cannot be accepting (rejecting); McNaughton [14] calls such a condition as lacking splits. He showed that in such a case the state set of the automaton can be partitioned into two sets, one which from player I has a memory-less winning strategy and the other from which player II has a memory-less winning strategy.

Our results bound the amount of memory needed for a strategy for player I (II) in a game presented as a DSA (DRA) as a function of the Rabin Index (k) of the language ($n.h^k$, where n and h are respectively the number of states and pairs in the GA).

7 The Rabin Index and real systems

In this section we discuss the relevance of the Rabin Index to the complexity of practical problems in synthesis and verification.

The Synthesis problem is equivalent to finding a regular tree in a tree automaton. There exist algorithms to check if an RTA or STA has nonempty language that are exponential in the number of pairs, with complexity $(nh)^{O(h)}$, where n is the number of states and h the number of pairs [4, 15]. On the other hand, nonemptiness of Chain TA is exponential only in the RI k (of complexity $n^{O(k)}$ [8]), and Buchi TA nonemptiness is in polynomial time.

In practice, hardware designs and protocols are conceived of a system of interacting components. Each component is typically simple and small, and has associated with it a simple fairness constraints such as Buchi or Co-Buchi, each expressible with a single Streett-pair. Although the product automaton is specified with as many pairs as the sum of the number of pairs in the components, often the Streett Index of the product automaton is smaller.

The constructions presented to convert Streett and Rabin automata to Chain automata can be employed in conjunction with the CTA-emptiness algorithm to yield a more efficient test for STA or RTA nonemptiness in case the corresponding game languages have low RI in comparison to the number of pairs. Therefore the controller-synthesis problem for a compositionally specified plant can be decided more efficiently for systems of low RI.

The constructions also have applications to testing language emptiness of a set of coordinating automata [12] that arises in the context of verification. As the product automaton is formed incrementally, by including more component automata, the number of pairs may be minimized. However, this is beneficial only when the SI is less than 2, since Streett ω-automata emptiness is quadratic in the number of pairs (as opposed to exponential for tree automata).

8 Conclusion

Our main contribution in this paper is a construction to convert a deterministic Rabin automaton with n states and h pairs into a deterministic Chain automaton with nh^k states, where k is the Rabin Index of the language. The result has several interesting applications to problems in ω-automata, tree automata, two-person games and controller synthesis. Chain automata can be trivially translated into minimum pair Rabin automata or Streett automata. Hence, our construction translates a DRA or DSA into either a minimum-pair DRA, or a minimum-pair DSA of size nh^k. The construction can be applied to a trim deterministic Rabin tree automaton to obtain either a minimum-pair deterministic Rabin tree automaton, or a deterministic Streett tree automaton of size nh^k, where k is the Rabin Index of the tree language.

The result also has several interesting implications for two-person infinite games, and the controller synthesis problem. The complexity of determining the winner and the strategy for the winner in a game given as a DRA or DSA is exponential in the number of pairs. Our result suggests the complexity should be a function of the Rabin Index of the game language. We also show that an upper bound on the amount of memory needed to implement a strategy for player I (II) in a game presented as a DSA (DRA) is nh^k, where k is the Rabin Index of the game language.

We are currently trying to devise an algorithm for Rabin tree automata nonemptiness that is exponential in the RI and is no worse than that in [4, 15] even in the worst case.

References

1. O. Carton. Chain Automata. In *IFIP 13th World Computer Congress*, pages 451–458, August 1994.
2. E. S. Chang, Z. Manna, and A. Pnueli. The Safety-Progress Classification. In F. L. Bauer, W. Bauer, and H. Schwichtenberg, editors, *Logic and Algebra of Specification*, pages 143–202, 1993.
3. E. A. Emerson. Automata, tableaux, and temporal logics. In *Logics of Programs*, LNCS, pages 79–88. Springer-Verlag, 1985.
4. E. A. Emerson and C. S. Jutla. The complexity of tree automata and logics of programs. In *Proc. of the Symp. on Foundations of Computer Science*, pages 328–337, October 1988.
5. E. A. Emerson and C. S. Jutla. Trees automata, Mu-calculus and determinacy. In *Proc. of the Symp. on Foundations of Computer Science*, pages 368–377, October 1991.
6. E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model-checking for fragments of mu -calculus. In *Computer Aided Verification*, volume 697 of *LNCS*, pages 385–396. Springer-Verlag, 1994.
7. Y. Gurevich and L. Harrington. Trees, automata, and games. In *Proc. of the ACM Symposium on the Theory of Computing*, pages 60–65, May 1982.
8. C. S. Jutla. Personal communication, February 1995.
9. M. Kaminski. A Classification of ω-regular languages. *Theoretical Computer Science*, 36:217–229, 1985.
10. S. C. Krishnan, A. Puri, and R. K. Brayton. Deterministic ω-automata vis-a-vis Deterministic Buchi Automata. In *Algorithms and Computation*, volume 834 of *LNCS*, pages 378–386. Springer-Verlag, 1994.
11. S. C. Krishnan, A. Puri, and R. K. Brayton. Structural Complexity of ω-automata. In *Symposium on Theoretical Aspects of Computer Science*, volume 900 of *LNCS*, pages 143–156. Springer-Verlag, 1995.
12. R. P. Kurshan. *Computer-aided Verification of Coordinating Processes: the Automata-theoretic approach.* Princeton University Press, 1994.
13. O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of Discrete Controllers for Timed Systems. In *Symposium on Theoretical Aspects of Computer Science*, volume 900 of *LNCS*, pages 229–242. Springer-Verlag, 1995.
14. R. McNaughton. Infinite gmaes played on finite graphs. *Annals of Pure and Applied Logic*, 65:149–184, 1993.
15. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. of the ACM Symposium on Principles of Programming Languages*, pages 179–180, 1989.
16. M. O. Rabin. *Automata on Infinite Objects and Church's Problem*, volume 13 of *Regional Conf. Series in Mathematics.* American Mathematical Society, Providence, Rhode Island, 1972.
17. S. Safra and M. Y. Vardi. On ω-Automata and Temporal Logic. In *Proc. of the ACM Symposium on the Theory of Computing*, pages 127–137, May 1989.
18. Shmuel Safra. *Complexity of Automata on Infinite Objects.* PhD thesis, The Weizmann Institute of Science, Rehovot, Israel, March 1989.
19. W. Thomas. Automata on Infinite Objects. In J. van Leeuwen, editor, *Formal Models and Semantics*, volume B of *Handbook of Theoretical Computer Science*, pages 133–191. Elsevier Science, 1990.
20. W. Thomas. On the synthesis of strategies in infinite games. In *Symposium on Theoretical Aspects of Computer Science*, volume 900 of *LNCS*, pages 1–13. Springer-Verlag, 1995.
21. K. Wagner. On ω-Regular Sets. *Information and Control*, 43:123–177, 1979.