12 Automata-theoretic Liveness Verification

12.1 $\omega$-Automata .............................................. 1
12.2 Operations on $\omega$-automata ................................. 7
  12.2.1 Product ................................................. 8
  12.2.2 Complementation ......................................... 10
12.3 Expressiveness .................................................. 16
  12.3.1 $\omega$-regular languages .................................. 16
  12.3.2 Expressiveness of $\omega$-automata ....................... 18
  12.3.3 Deterministic $\omega$-automata ............................ 19
Chapter 12

Automata-theoretic Liveness Verification

In this chapter, we extend the automata-theoretic approach studied in Chapter 8 for safety requirements to liveness requirements. In the automata-theoretic liveness verification, a fair module is viewed as a generator of an $\omega$-language, namely, the set of its fair traces, the requirement is specified by an $\omega$-automaton that accepts only the desired $\omega$-traces, and the verification problem corresponds to inclusion between these two $\omega$-languages.

12.1 $\omega$-Automata

A fair structure $K$ consists of an observation structure $K$ and a fairness assumption $F$. Each fair structure defines the $\omega$-language $\mathcal{L}_K$ over the set of its observations consisting of the set of $\omega$-traces corresponding to initialized $F$-fair $\omega$-trajectories. Fair structures can be used to specify requirements also. In their role as a specification language, fairness constraints are usually specified using regions rather than actions. In this role, fairness constraints should be viewed as accepting conditions that classify $\omega$-trajectories into accepting and rejecting rather than assumptions about fair resolution of choice. We will concentrate on two types of accepting conditions: Büchi acceptance and Streett acceptance.
Büchi automata

Finite structures with a single weak-fairness constraint specified by a region are called Büchi automata.

**Büchi automaton**

A Büchi automaton \( M \) consists of (1) a finite observation structure \( K \) and (2) a region \( \sigma^A \) of \( K \). An initialized \( \omega \)-trajectory \( \gamma \) of \( K \) is accepted by the Büchi automaton \( M \) if \( s_i \in \sigma^A \) for infinitely many positions \( i \geq 0 \). The \( \omega \)-language \( L_M \) of the Büchi automaton \( M \) is the set of traces corresponding to initialized accepted trajectories of \( M \). The Büchi automaton \((K, \sigma^A)\) is deterministic if the observation structure \( K \) is deterministic.

Note that syntactically a Büchi automaton is identical to an ordinary automaton. In an ordinary automaton, a (finite) trajectory is accepted if it terminates in an accepting state; in a Büchi automaton, an \( \omega \)-trajectory is accepted if its visits a repeating state infinitely often.

**Example 12.1** [Büchi languages] Let \( A = \{a, b\} \). The Büchi automaton \( M_1 \) of Figure 12.1 accepts the response language \((b^*a)\) consisting of \( \omega \)-words that contain infinitely many \( a \) symbols. The Büchi automaton \( M_2 \) of Figure 12.1 accepts the persistence language \( A^*a^\omega \) consisting of \( \omega \)-words with a suffix containing only \( a \) symbols. Note that the automaton \( M_2 \) is nondeterministic (it guesses the beginning of the suffix containing only \( a \) symbols).

Let \( A = \{a, b, c\} \). The nondeterministic Büchi automaton \( M_3 \) of Figure 12.1 accepts the reactivity language consisting of \( \omega \)-words that contain either only finitely many \( a \) symbols or infinitely many \( b \) symbols. Note that “finitely many \( a \) or infinitely many \( b \)” is equivalent to “infinitely many \( b \) or eventually always \( c \).” ■

**Remark 12.1** [Multi-Büchi automaton] A multi-Büchi automaton \( M \) consists of (1) a finite observation structure \( K \), and (2) a finite set \( F \) of repeating regions of \( K \). An initialized \( \omega \)-trajectory \( \gamma \) of \( K \) is accepted by the multi-Büchi automaton \( M \) if for every repeating region \( \sigma \in F \), \( s_i \in \sigma \) for infinitely many positions \( i \geq 0 \). Thus, a multi-Büchi automaton is a weak-fair structure all of whose weak-fairness constraints are specified by regions. ■

**Exercise 12.1** [P2] [CoBüchi automata] A CoBüchi automaton \( M \) consists of (1) a finite observation structure \( K \) and (2) a region \( \sigma^A \) of \( K \). An initialized \( \omega \)-trajectory \( \gamma \) of \( K \) is accepted by the CoBüchi automaton \( M \) if it has a suffix all of whose states are in the stable region: there exists \( i \geq 0 \) such that \( s_j \in \sigma^A \) for all \( j \geq i \). Note that syntactically a CoBüchi automaton is
Automata-theoretic Liveness Verification

like a Büchi automaton or an ordinary automaton. Semantically, the CoBüchi automaton \((K, \sigma^A)\) is like the fair structure \((K, \{\iota \sigma^A, \emptyset\})\).

Consider the alphabet \(A = \{a, b\}\). (1) Find a CoBüchi automaton whose \(\omega\)-language is the persistence language \(A^*a^\omega\) consisting \(\omega\)-words with a suffix containing only \(a\) symbols. (2) Consider the response language \((b^*a)^\omega\) consisting of \(\omega\)-words with infinitely many \(a\) symbols. Can you draw a CoBüchi automaton that accepts this language? ■

**Streett Automata**

Finite structures with a fairness constraints specified by regions are called Streett automata.
Automata-theoretic Liveness Verification

A Streett automaton $\mathcal{M}$ consists of (1) a finite observation structure $K$ and (2) [the Streett constraints] a finite set $F$ of pairs of regions. An initialized $\omega$-trajectory $s$ of $K$ is accepted by the Streett automaton $\mathcal{M}$ if for every Streett constraint $(\sigma, \tau) \in F$, if $s$ is $\sigma$-fair then $s$ is $\tau$-fair. The $\omega$-language $L_{\mathcal{M}}$ of the Streett automaton $\mathcal{M}$ is the set of $\omega$-traces corresponding to initialized accepted $\omega$-trajectories of $\mathcal{M}$.

Remark 12.2 [Büchi as a special case of Streett] A Büchi automaton $(K, \sigma^A)$ can be viewed as the Streett automaton $(K, \{\{\Sigma, \sigma^A\}\})$ with a single Streett constraint.

Example 12.2 [Streett language] Let $A = \{a, b, c\}$. The nondeterministic Büchi automaton $\mathcal{M}_3$ of Figure 12.1 accepts the reactivity language consisting of $\omega$-words that contain either only finitely many $a$ symbols or infinitely many $b$ symbols. The same $\omega$-language is accepted by the deterministic Streett automaton $\mathcal{M}$ of Figure 12.2. There is a single Streett constraint $(\{s\}, \{t\})$.

Exercise 12.2 [P2] [Rabin automata] A Rabin automaton $\mathcal{M}$ is syntactically identical to a Streett automaton, and consists of (1) a finite observation structure $K$ and (2) [the Rabin constraints] a finite set $F$ of pairs of regions. An initialized $\omega$-trajectory $s$ of $K$ is accepted by the Rabin automaton $\mathcal{M}$ if there exists a Rabin constraint $(\sigma, \tau) \in F$ such that $s$ is $\sigma$-fair and but not $\tau$-fair. Thus,
Automata-theoretic Liveness Verification

semantically a Rabin automaton is the dual of the Streett automaton: Streett acceptance has the form

$$\bigwedge (\sigma, \tau) \in F. \ [(\neg \sigma \text{-fair}) \lor \tau \text{-fair}] ,$$

while Rabin acceptance requires

$$\bigvee (\sigma, \tau) \in F. \ [\sigma \text{-fair} \land \neg (\tau \text{-fair})] .$$

Let $$A = \{a, b, c\}$$. Find a deterministic Rabin automaton $$M$$ accepting the reactivity language consisting of $$\omega$$-words that contain either only finitely many $$a$$ symbols or infinitely many $$b$$ symbols.

Exercise 12.3 \{T3\} [Muller automata] A Muller automaton $$M$$ is syntactically like a multi-Büchi automaton, and consists of (1) a finite observation structure $$K$$ and (2) [the Muller acceptance] a finite set $$F$$ of regions of $$K$$. An initialized $$\omega$$-trajectory $$s$$ of $$K$$ is accepted by the Muller automaton $$M$$ if the set $$\{s \in \Sigma \mid s_i = s \text{ for infinitely many } i \geq 0\}$$ of states repeating infinitely often along $$s$$ is in $$F$$. Show that Streett automata as well as Rabin automata are special cases of Muller automata.

The $$\omega$$-language-inclusion problem

The $$\omega$$-language-inclusion problem asks whether every $$\omega$$-trace accepted by one $$\omega$$-automaton is also accepted by another $$\omega$$-automaton.

<table>
<thead>
<tr>
<th>The $$\omega$$-Language-Inclusion Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>An instance $$(M_1, M_2)$$ of the $$\omega$$-language-inclusion problem consists of two $\omega$-automata $$M_1$$ and $$M_2$$ over the same observation alphabet $$A$$. The answer to the $$\omega$$-language-inclusion problem $$(M_1, M_2)$$ is Yes if $$L_{M_1} \subseteq L_{M_2}$$, and otherwise No.</td>
</tr>
</tbody>
</table>

Note that in an instance $$(M_1, M_2)$$ of the $$\omega$$-language-inclusion problem, each of the $$\omega$$-automata $$M_1$$ and $$M_2$$ may be either a fair structure, or a Büchi automaton, or a Streett automaton.

Automata as specifications

$$\omega$$-automata can be used for specifying requirements of fair modules. As in case of the logic SAL, the observations of the requirements automaton are boolean expressions over the observable variables of modules. We define the fair state logic LAL whose formulas are Büchi and Streett automata.
LIVE AUTOMATON LOGIC
A formula of the fair state logic *live automaton logic* (LAL) is a Büchi or a Streett automaton $\mathcal{M}$ whose observations are boolean expressions.

Given a formula $\mathcal{M}$ of LAL, a fair structure $\mathcal{K}$ is a $\mathcal{M}$-structure if each observation of $\mathcal{K}$ is a valuation for a superset of the variables appearing in the observations of $\mathcal{M}$.

The satisfaction relation for LAL is defined by:

$$s \models_{\mathcal{K}} \mathcal{M} \iff \text{for every source-s fair } \omega\text{-trajectory } \mathfrak{a} \text{ of } \mathcal{K}$$

$$\text{there is an accepting } \omega\text{-trace } \mathfrak{a} \in \mathcal{L}_{\mathcal{M}} \text{ such that}$$

$$\text{for all } i \geq 0, s_i \models a_i.$$

In other words, a state $s$ of $\mathcal{K}$ satisfies the requirement specified by the $\omega$-automaton $\mathcal{M}$ if for every source-s fair $\omega$-trajectory $\mathfrak{a}$ of $\mathcal{K}$, we can find an initialized accepting $\omega$-trace $\mathfrak{a} \in \mathcal{L}_{\mathcal{M}}$ such that every observation in $\mathfrak{a}$ is consistent with the corresponding expression in $\mathfrak{a}$.

**Example 12.3** [LAL] The LAL formula $\mathcal{M}_{\mathcal{V}U}$ shown in Figure 12.3 asserts that, given a state $s$, every source-s fair $\omega$-trajectory contains a state satisfying $y$ which is preceded only by states satisfying $x$. The formula $\mathcal{M}_{\mathcal{V}U}$ can be interpreted at states of a fair structure whose observations assign values to $x$ and $y$. It follows that the LAL formula $\mathcal{M}_{\mathcal{V}U}$ is equivalent to the CTL formula $x\forall U y$. Contrast the specification $\mathcal{M}_{\mathcal{V}U}$ with the SAL specification $M_{\forall y}$ corresponding to the STL formula $x\forall W y$ (see Example 8.2).
Example 12.4 [Starvation freedom in live automaton logic] Recall the starvation freedom requirement for mutual exclusion protocols. The requirement that \( p_{c_1} \neq \text{reqC} \) be a recurrent is expressed in LAL by the Büchi automaton of Figure 12.4. ■

**LAL model checking**

The model-checking problem for LAL can be reduced to the \( \omega \)-language-inclusion problem. As in case of SAL, we expand each \( \omega \)-automaton \( \mathcal{M} \) of LAL to a larger automaton \( E\mathcal{M} \) whose observations are valuations to the variables appearing in the observations of \( \mathcal{M} \). Recall the definition of the expansion operator \( E \) from Chapter 8. To obtain expansion of an \( \omega \)-automaton, we apply the expansion operation to the underlying observation structure, and modify the accepting condition appropriately.

**Expansion of an LAL automaton**

For a Büchi automaton \( \mathcal{M} = (K, \sigma^A) \) given as a LAL formula, the expansion \( E\mathcal{M} \) is another Büchi automaton: (1) the observation structure of \( E\mathcal{M} \) is \( EK \), and (2) the repeating region of \( E\mathcal{M} \) is \( \sigma^A \uparrow \).

For a Streett automaton \( \mathcal{M} = (K, F) \) given as a LAL formula, the expansion \( E\mathcal{M} \) is another Streett automaton: (1) the observation structure of \( E\mathcal{M} \) is \( EK \), and (2) for every \((\sigma, \tau) \in F\), the automaton \( E\mathcal{M} \) has the Streett constraint \((\sigma \uparrow, \tau \uparrow)\).

**Exercise 12.4** \{P1\} [LAL expansion] Draw the expanded Büchi automaton corresponding to the LAL specification \( \mathcal{M}_{\text{util}} \) of Figure 12.3. ■

It follows that checking whether a fair structure satisfies an LAL automaton \( \mathcal{M} \) is equivalent to checking whether \( K \) satisfies the expanded \( \omega \)-automaton \( E\mathcal{M} \), which in turn corresponds to checking whether the fair language of \( K \) is contained in the fair language of \( E\mathcal{M} \).

**Proposition 12.1** [LAL model checking] The LAL model-checking problem \((K, \mathcal{M})\) and the \( \omega \)-language-inclusion problem \((K, E\mathcal{M})\) have the same answer.

### 12.2 Operations on \( \omega \)-automata

To solve the \( \omega \)-language inclusion problem \((\mathcal{M}_1, \mathcal{M}_2)\), we first obtain an \( \omega \)-automaton that accepts the complement of the \( \omega \)-language accepted by \( \mathcal{M}_2 \), then construct its product with \( \mathcal{M}_1 \), and solve the fair emptiness problem on the resulting \( \omega \)-automaton.
12.2.1 Product

Given two \( \omega \)-automata \( \mathcal{M}_1 \) and \( \mathcal{M}_2 \), we wish to define another \( \omega \)-automaton that accepts the intersection of the \( \omega \)-languages of \( \mathcal{M}_1 \) and \( \mathcal{M}_2 \). For this purpose, we resort to the product construction described in Section 8.3.2 over observation structures. Consider two observation structures \( K_1 \) and \( K_2 \), and let \( K_1 \times K_2 \) be their product. Let \( s \) be an \( \omega \)-trajectory of the product. Then, by the definition of the product, there exists an \( \omega \)-trajectory \( t \) of \( K_1 \) and an \( \omega \)-trajectory \( u \) of \( K_2 \) such that \( s = (t_0, u_0)(t_1, u_1) \cdots \). Observe that, for a region \( \sigma \) of \( K_1 \), the \( \omega \)-trajectory \( t \) of \( K_1 \) is \( \sigma \)-fair iff the \( \omega \)-trajectory \( s \) of the product is \((\sigma \uparrow)\)-fair. Similarly, for a region \( \sigma \) of \( K_2 \), the \( \omega \)-trajectory \( u \) of \( K_2 \) is \( \sigma \)-fair iff the \( \omega \)-trajectory \( s \) of the product is \((\sigma \uparrow)\)-fair. In other words, fairness with respect to a region \( \sigma \) in a component translates to fairness with respect to the lifted region \( \sigma \uparrow \) in the product. Similarly, fairness with respect to an action \( \alpha \) in a component translates to fairness with respect to the lifted action \( \alpha \uparrow \) in the product. This leads to a natural definition of product for \( \omega \)-automata.

**Product of \( \omega \)-automata**

Let \( \mathcal{M}_1 = (K_1, F_1) \) and \( \mathcal{M}_2 = (K_2, F_2) \) be two Streett automata. The product \( \mathcal{M}_1 \times \mathcal{M}_2 \) is the Streett automaton \( (K_1 \times K_2, \{(\sigma \uparrow, \tau \uparrow) \mid (\sigma, \tau) \in F_1 \cup F_2\}) \).

Let \( K_1 = (K_1, F_1) \) and \( K_2 = (K_2, F_2) \) be two fair structures. The product \( K_1 \times K_2 \) is the fair structure \( (K_1 \times K_2, \{(\alpha \uparrow, \beta \uparrow) \mid (\alpha, \beta) \in F_1 \cup F_2\}) \).

**Proposition 12.2** [Product of \( \omega \)-automata] If \( \mathcal{M}_1 \) and \( \mathcal{M}_2 \) are two Streett automata, then \( L_{\mathcal{M}_1 \times \mathcal{M}_2} = L_{\mathcal{M}_1} \cap L_{\mathcal{M}_2} \). If \( K_1 \) and \( K_2 \) are two fair structures, then \( L_{K_1 \times K_2} = L_{K_1} \cap L_{K_2} \).

**Remark 12.3** [Cost of product] Let \( \mathcal{M}_1 \) be a Streett automaton with \( n_1 \) states, \( m_1 \) transitions, and \( \ell_1 \) Streett constraints. Let \( \mathcal{M}_2 \) be a Streett automaton with \( n_2 \) states, \( m_2 \) transitions, and \( \ell_2 \) Streett constraints. Then, the product \( \mathcal{M}_1 \times \mathcal{M}_2 \) has at most \( n_1 \cdot n_2 \) states, at most \( m_1 \cdot m_2 \) transitions, and \( \ell_1 + \ell_2 \) Streett constraints.

**Product of Buchi automata**

The product of two Buchi automata \( (K_1, \sigma_1^A) \) and \( (K_2, \sigma_2^A) \) can be defined to be the multi-Buchi automaton \( (K_1 \times K_2, \{\sigma_1^A \uparrow, \sigma_2^A \uparrow\}) \). However, by introducing a counter, as described in Section 11.3.2, we can define product of Buchi automata to be a Buchi automaton. The states of the product are, then, of the form \((s, t, i)\), where \( s \) is a state of \( K_1 \), \( t \) is a state of \( K_2 \), and \( i \) is a counter that can be either 1 or 2. The counter is updated from 1 to 2 when an accepting state of \( K_1 \) is visited, and from 2 to 1 when an accepting state of \( K_2 \) is visited. The
accepting condition of the product requires infinitely many updates from 2 to 1.

**Proposition 12.3** [Product of Büchi automata] If $M_1$ and $M_2$ are two Büchi automata, then $L_{M_1 \times M_2} = L_{M_1} \cap L_{M_2}$. 
Example 12.5 [Product of Büchi automata] Consider the two Büchi automata $M_1$ and $M_2$ of Figure 12.5. The automaton $M_1$ accepts all $\omega$-words that contain infinitely many $a$ symbols, while $M_2$ accepts all $\omega$-words that contain infinitely many $b$ symbols. The result of applying the product construction contains 4 states, of which the only accepting state is $(t, v, 2)$. Verify that the product accepts precise those $\omega$-words that contain infinitely many $a$ symbols as well as infinitely many $b$ symbols.

Remark 12.4 [Product of deterministic Büchi automata] If $M_1$ and $M_2$ are deterministic Büchi automata, then so is their product $M_1 \times M_2$. Thus, the class of $\omega$-languages definable by deterministic Büchi automata is closed under intersection.

Exercise 12.5 [T2] [Product of CoBüchi automata] Given two CoBüchi automata $M_1$ and $M_2$, define a CoBüchi automaton $M_1 \times M_2$ that accepts the intersection of the $\omega$-languages of $M_1$ and $M_2$.

12.2.2 Complementation

We turn our attention to the problem of complementing a Büchi automaton. Recall that for an ordinary automaton, its complement is constructed by first determinizing the observation structure using the subset construction, followed by completion by adding dummy states, followed by inversion of the accepting condition. Given a Büchi automaton $M = (K, \sigma^A)$, can we add accepting conditions to the determinized structure $\Delta K$ without changing the $\omega$-language accepted by $M$? The obstacle in such an approach is illustrated by the following example.

Example 12.6 [Subset construction and Büchi automata] Recall the Büchi automaton $M_2$ from Figure 12.1 that accepts the persistence language $A^*a^\omega$. The
The problem with the subset construction is that states of the determinized structure may contain both repeating and nonrepeating states. Complementing a nondeterministic Büchi automaton turns out to be a nontrivial problem. Consequently, existing model checkers do not support nondeterministic $\omega$-automata as specifications. However, understanding the complementation procedure provides insights into the structure of $\omega$-automata.

We begin by some preliminary definitions. Let $A$ be a finite alphabet. An equivalence relation $\sim \subseteq A^* \times A^*$ over words over $A$ is said to be a congruence (with respect to concatenation) if for all words $\vec{a}, \vec{b},$ and $\vec{c},$ if $\vec{a} \sim \vec{b}$ then $\vec{a} \cdot \vec{c} \sim \vec{b} \cdot \vec{c}$ and $\vec{a} \cdot \vec{c} \sim \vec{a} \cdot \vec{b}$. By a finite equivalence relation, we mean an equivalence relation with finitely many equivalence classes.

Let $M = (\Sigma, \sigma^I, \rightarrow, A, \langle \langle \cdot \rangle \rangle, \sigma^A)$ be a Büchi automaton. We are going to establish that both $L_M$ and its complement can be expressed as finite unions of $\omega$-languages of the form $L_1 \cdot L_2^\omega$, where $L_1$ and $L_2$ are blocks of a certain finite congruence on $A^*$.

For two states $s$ and $t$ of $M$, and a word $\vec{a}_0 \ldots \vec{a}_m$ over $A$, define $s \alpha^T \vec{a} t$ if there is a trajectory $s_0 \ldots s_m$ of $M$ such that $s_0 = s$, $s_m = t$, and $\langle s_i \rangle = a_i$ for all $0 \leq i \leq m$.

That is, $s \alpha^T \vec{a} t$ means that the trace $\vec{a}$ can lead the automaton from the initial state $s$ to the final state $t$.

For two states $s$ and $t$ of $M$, and a word $\vec{a}_0 \ldots \vec{a}_m$ over $A$, define $s \alpha^T \vec{a} t$ if there is a trajectory $s_0 \ldots s_m$ of $M$ such that $s_0 = s$, $s_m = t$, $\langle s_i \rangle = a_i$ for all $0 \leq i \leq m$, and $s_j \in \sigma^A$ for some $0 \leq j \leq m$.

That is, $s \alpha^T \vec{a} t$ means that the trace $\vec{a}$ can lead the automaton from the initial state $s$ to the final state $t$ via a trajectory that visits some repeating state. Now we are ready to define the desired equivalence relation on $A^*$ induced by $M$:

For two words $\vec{a}$ and $\vec{b}$ over $A$, $\vec{a} \sim_M \vec{b}$ iff for all states $s$ and $t$ of $M$, (1) $s \alpha^T \vec{a} t$ iff $s \alpha^T \vec{b} t$, and (2) $s \alpha^T \vec{a} t$ iff $s \alpha^T \vec{b} t$.

First, we note that the equivalence $\sim_M$ is a finite congruence:

Lemma 12.1 [Congruence] The equivalence relation $\sim_M$ over $A^*$ is a congruence with respect to concatenation.

Proof. Left as an exercise.
Lemma 12.2 [Finiteness] The equivalence relation \( \sim_M \) over \( A^* \) is finite, and has at most \( 2^{2n^2} \) classes if \( M \) has \( n \) states.

Proof. For every two states \( s \) and \( t \) of \( M \), let \( L_{s,t} \) be the language containing words \( \overline{a} \) such that \( s \alpha T \overline{a} t \), and let \( L'_{s,t} \) be the language containing words \( \overline{a} \) such that \( s \alpha T \overline{a} t \). Let \( \Pi \) be the set of these \( 2n^2 \) languages. Now an equivalence class of \( \sim_M \) corresponds to a subset of \( \Pi \): given a subset \( \Pi' \subseteq \Pi \), the intersection

\[
[ \bigcap L \in \Pi'. L ] \cap [ \bigcap L \notin \Pi'. A^* \setminus L ]
\]

defines an equivalence class of \( \sim_M \). It follows that the number of subsets of \( \Pi \) is an upper bound for the number of equivalence classes of \( \sim_M \). \( \blacksquare \)

The next lemma asserts a saturation property of the \( \sim_M \)-equivalence classes with respect to the \( \omega \)-language accepted by \( M \):

Lemma 12.3 [Saturation] Let \( L_1 \) and \( L_2 \) be two equivalence classes of the congruence \( \sim_M \). Then, if \( L_1 \cdot L_2^c \cap \mathcal{L}_M \) is nonempty then \( L_1 \cdot L_2^c \subseteq \mathcal{L}_M \).

Proof. Let \( L_1 \) and \( L_2 \) be two equivalence classes of \( \sim_M \). Suppose \( L_1 \cdot L_2^c \cap \mathcal{L}_M \) is nonempty, and contains the \( \omega \)-word \( \overline{a} \). Since \( \overline{a} \in L_1 \cdot L_2^c \), it is of the form \( \overline{b}_0 \cdot \overline{b}_1 \cdot \overline{b}_2 \cdots \), where the word \( \overline{b}_0 \) is in \( L_1 \) and for \( i \geq 1 \), the word \( \overline{b}_i \) is in \( L_2 \). Since \( \overline{a} \) is accepted by \( M \), there exists an initialized accepting \( \omega \)-trajectory corresponding to \( \overline{a} \). Thus, there exist states \( s_0, s_1, \ldots \) such that

\[
s_0 \alpha T \overline{b}_0 \ s_1 \alpha T \overline{b}_1 \ s_1 \alpha T \overline{b}_2 \cdots
\]

Furthermore, for infinitely many indices \( i, s_i \alpha T \overline{b}_i, s_{i+1} \).

Now consider another word \( \overline{c} \in L_1 \cdot L_2^c \). We need to establish that \( M \) accepts \( \overline{c} \) also. The \( \omega \)-word \( \overline{c} \) is of the form \( \overline{d}_0 \cdot \overline{d}_1 \cdot \overline{d}_2 \cdots \) such that the word \( \overline{d}_0 \) is in \( L_1 \) and for \( i \geq 1 \), the word \( \overline{d}_i \) is in \( L_2 \). Since \( L_1 \) and \( L_2 \) are equivalence classes of \( \sim_M \), \( \overline{b}_i \sim_M \overline{d}_i \) for all \( i \geq 0 \). It follows that

\[
s_0 \alpha T \overline{d}_0 \ s_1 \alpha T \overline{d}_1 \ s_1 \alpha T \overline{d}_2 \cdots
\]

and for infinitely many indices \( i, s_i \alpha T \overline{d}_i, s_{i+1} \). We conclude that there is an initialized accepting trajectory corresponding to the \( \omega \)-word \( \overline{c} \). \( \blacksquare \)

The next lemma asserts that \( \omega \)-languages of the form \( L_1 \cdot L_2^c \) cover the set of all \( \omega \)-words, provided \( L_1, L_2 \) range over equivalence classes of a finite congruence.

Lemma 12.4 [Coverage] Let \( \sim \) be a finite congruence over \( A^* \), and let \( \overline{a} \) be an \( \omega \)-word over \( A \). Then, there exist equivalence classes \( L_1 \) and \( L_2 \) of \( \sim \) such that \( \overline{a} \in L_1 \cdot L_2^c \).
Proof. Let \( \sim \) be a finite congruence relation over \( A^* \), and let \( g \) be an infinite word over \( A \). We say that two indices \( i \) and \( j \) merge at an index \( k > i, j \) if \( \pi_{i..k} \sim \pi_{j..k} \). For two indices \( i \) and \( j \), define \( i \equiv j \) if they merge at some index. Verify that \( \equiv \) is an equivalence relation over the set of nonnegative integers. Furthermore, given a finite subset \( D \) of nonnegative integers such that \( D \) has more elements than the number of equivalence classes of \( \sim \), if we choose \( k \) such that \( k > i \) for all \( i \in D \), then the set \( \{ \pi_{i..k} \mid i \in D \} \) must contain two \( \sim \)-equivalent words. It follows that the equivalence relation \( \equiv \) itself is finite (the number of equivalence classes of \( \equiv \) is bounded by the number of equivalence classes of \( \sim \)).

Finiteness of \( \equiv \) implies that there exists an infinite sequence \( i_0 < i_1 < i_2 < \cdots \) of indices that are \( \equiv \)-equivalent to each other. Note that for every \( j \geq 1 \), all the indices \( i_0, i_1, \ldots, i_j \) merge at some \( k > i_j \). Without loss of generality, we may assume that for every \( j \geq 0 \), all the indices \( i_0, i_1, \ldots, i_j \) merge at \( i_{j+1} \) (this is because we can delete indices from the original sequence, and if \( i_0, i_1, \ldots, i_j \) merge at \( k \) then they merge at every \( k' > k \) as \( \sim \) is a congruence). It follows that there is an infinite sequence \( i_0, i_1, i_2, \ldots \) of indices such that

1. all the words in \( \{ \pi_{i_0..i_j} \mid j \geq 1 \} \) belong to the same equivalence class of \( \sim \), let this class be \( L_2 \).

2. for all \( j < j' < k \), the indices \( i_j \) and \( i_{j'} \) merge at \( i_k \).

From (1), \( \pi_{i_0..i_j} \) is in \( L_2 \). For all \( j \geq 1 \), \( \pi_{i_0..i_{j+1}} \) is in \( L_2 \) by (1), and \( \pi_{i_0..i_{j+1}} \) is \( \sim \)-equivalent to \( \pi_{i_j..i_{j+1}} \) by (2). It follows that for all \( j \geq 0 \), \( \pi_{i_j..i_{j+1}} \) is in \( L_2 \).

It follows that the suffix \( \pi_{i_0..} \) is in \( L_2^\omega \). This completes the proof. \( \blacksquare \)

Since \( \sim_M \) is a finite congruence, it follows that the set
\[
\{ L_1 \cdot L_2^\omega \mid L_1, L_2 \text{ are equivalence classes of } \sim_M \}
\]
covers \( A^\omega \), and then, by the saturation property, the \( \omega \)-language accepted by \( \mathcal{M} \) corresponds to a subset of this set, and the complement defines the complementary language.

Corollary 12.1 [Structure of Büchi language] The \( \omega \)-language \( L_M \) accepted by the Büchi automaton \( \mathcal{M} \) equals
\[
\bigcup \{ L_1 \cdot L_2^\omega \mid L_1, L_2 \text{ are equivalence classes of } \sim_M \text{ and } L_1 \cdot L_2^\omega \cap L_M \neq \emptyset \},
\]
and the complementary \( \omega \)-language \( A^\omega \setminus L_M \) equals
\[
\bigcup \{ L_1 \cdot L_2^\omega \mid L_1, L_2 \text{ are equivalence classes of } \sim_M \text{ and } L_1 \cdot L_2^\omega \cap L_M = \emptyset \}.
\]

Proof. Follows from Lemmas 12.1, 12.2, 12.3, and 12.4. \( \blacksquare \)

The next proposition asserts that if \( L_1 \) and \( L_2 \) are two regular languages then the \( \omega \)-language \( L_1 \cdot L_2^\omega \) is accepted by a Büchi automaton.
Proposition 12.4 [Regular concatenation] If $L_1$ and $L_2$ are two regular languages over $A$ then the $\omega$-language $L_1 \cdot L_2^\omega$ is accepted by some Büchi automaton.

Proof. Let $L_1$ be a regular language accepted by the automaton $M_1 = (\Sigma_1, \sigma_1^1, \rightarrow_1, A, \langle\rangle_1, \sigma_1^A)$, and let $L_2$ be a regular language accepted by the automaton $M_2 = (\Sigma_2, \sigma_2^2, \rightarrow_2, A, \langle\rangle_2, \sigma_2^A)$. The $\omega$-automaton accepting $L_1 \cdot L_2^\omega$ is obtained by taking disjoint union of the two automata $M_1$ and $M_2$, and adding transitions from accepting states of $M_1$ to the initial states of $M_2$, and from accepting states of $M_2$ to the initial states of $M_2$. Specifically, define the Büchi automaton $M$ over the alphabet $M$:

1. the state-space of $M$ is $\Sigma_1 \cup \Sigma_2$ (assuming $\Sigma_1$ and $\Sigma_2$ are disjoint sets),
2. the initial region of $M$ is $\langle\rangle_1$, 
3. the set of transitions of $M$ equals $\rightarrow_1 \cup \rightarrow_2 \cup (\sigma_1^A \times \sigma_2^A)$, 
4. the observation of a state $s$ of $M$ is $\langle s \rangle_1$ if $s \in \Sigma_1$ and $\langle s \rangle_2$ otherwise, 
5. the repeating region of $M$ is $\sigma_2^A$. 

Verify that an $\omega$-word $a$ is accepted by the Büchi automaton $M$ precisely when it belongs to $L_1 \cdot L_2^\omega$. □

Remark 12.5 [Regular concatenation] If the regular language $L_1$ is accepted by an automaton with $n_1$ states, and the regular language $L_2$ is accepted by an automaton with $n_2$ states, then $L_1 \cdot L_2^\omega$ is accepted by a Büchi automaton with $n_1 + n_2$ states. □

Theorem 12.1 [Büchi Theorem on Complementation Closure] Given a Büchi automaton $M$, there exists a Büchi automaton $M'$ such that $L_{M'} = A^\omega \setminus L_M$.

Proof. Let $M$ be a Büchi automaton. The languages $L_{s,t}$ and $L'_{s,t}$ defined in the proof of Lemma 12.2 are regular languages. Since regular languages are closed under complement and intersection, from the proof of Lemma 12.2 it follows that every equivalence class of $\sim_M$ is a regular language. By Proposition 12.4, for two equivalence classes $L_1$ and $L_2$ of $\sim_M$, the $\omega$-language $L_1 \cdot L_2^\omega$ is accepted by a Büchi automaton. Since Büchi automata are closed under union, from Corollary 12.1, it follows that $A^\omega \setminus L_M$ is accepted by some Büchi automaton. □

Complexity of Complementation

Let us now analyze the bound on the size of the Büchi automaton accepting the complement of $L_M$ obtained by our construction. Suppose $M$ has $n$ states. Recall that $L_{s,t}$, for two states $s$ and $t$ of $M$, is the language containing words $\sigma$ such that $s \sigma T \pi t$. It follows that there an automaton accepting $L_{s,t}$ with $n$ states (use the same states and transitions of $M$, but declare $s$ to be initial and $t$ to be accepting). The language $L'_{s,t}$ containing words $\sigma$ such that $s \sigma T \pi t$ can be accepted by an automaton with $2n$ states (the state-space of $M$ is doubled to keep track of whether an accepting state has been visited or not). It follows that the language $A^\omega \setminus L_{s,t}$ is accepted by an automaton with $2^n$ states, and
the language $A^* \setminus L_{s,t}$ is accepted by an automaton with $4^n$ states (complementation may require determinization). Recall that an equivalence class of $\sim_M$ corresponds to a subset $\Pi'$ of the set $\Pi$ containing $2n^2$ languages $L_{s,t}, L_{s',t'}$. Such an equivalence class $\Pi'$ is defined by the product of the automata accepting $L$ for $L \in \Pi'$ and the automata accepting $A^* \setminus L$ for $L \not\in \Pi'$. Consequently, the bound on the size of the automaton accepting an equivalence class of $\sim_M$ is $(2^n)^n \cdot (4^n)^{n^2}$, which equals $2^{3n^3}$.

The number of states of the Büchi automaton accepting $L_1 \cdot L_2'$ equals the sum of the number of states of the automata accepting $L_1$ and $L_2$. Thus, for two equivalence classes $L_1$ and $L_2$ of $\sim_M$, there is a Büchi automaton with $2^{3n^3} + 1$ states accepting $L_1 \cdot L_2'$.

The number of states of the Büchi automaton accepting the union of $\omega$-languages equals the sum of the number of states of the Büchi automata accepting the disjuncts. Since the number of pairs of equivalence classes of $\sim_M$ is $2^{4n^2}$, from Corollary 12.1, the next theorem follows.

**Theorem 12.2** [Complexity of Büchi complementation] Given a Büchi automaton $M$ with $n$ states, there exists a Büchi automaton $M'$ with $2^{3n^3} + 4n^2 + 1$ states such that $L_{M'} = A^* \setminus L_M$.

Note that to construct the desired complement automaton, we need to construct, for every pair $L_1$ and $L_2$ of equivalence classes of $\sim_M$, the Büchi automaton accepting $L_1 \cdot L_2'$, and check if it has a nonempty intersection with $L_M$. We have already outlined the product construction to obtain intersection of the languages accepted by two Büchi automata. Algorithms for checking fair cycles from Chapter 10 can be used to check for emptiness.

**Remark 12.6** [Safra’s Construction] The complementation construction presented here yields an automaton with $2^{3n^3} + 4n^2 + 1$ states. Better constructions are known. In particular, Safra’s complementation algorithm produces an automaton with $2^{O(n \log n)}$ states. This is essentially optimal: $2^{n \log n}$ is a lower bound on the number of states necessary to define complements of a family of Büchi automata.

**Complementing Streett automata**

To establish that a Streett automaton can be complemented, we show that every Streett automaton has a language-equivalent Büchi automaton.

**Proposition 12.5** [From Streett to Büchi] Let $M$ be a Streett automaton over $A$. There exists a Büchi automaton $M'$ over $A$ such that $L_M = L_{M'}$.

**Proof.** Let $M = (\Sigma, \sigma', \rightarrow, A, \llbracket \cdot \rrbracket, F)$ be a Streett automaton. Recall that an $\omega$-trajectory $s$ is $F$-fair iff there exists a subset $F' \subseteq F$ of Streett constraints
and an index $i \geq 0$ such that (1) for every $(\sigma, \tau) \in F$, $s_i$ is $\tau$-fair, and (2) for every $(\sigma, \tau) \notin F$, $s_j \notin \sigma$ for all $j \geq i$.

Suppose $F$ has $\ell$ Streett constraints. The Büchi automaton $\mathcal{M}'$ has $2^\ell + 1$ copies of the observation structure of $\mathcal{M}$, an initial copy together with a copy for every subset $F' \subseteq F$ of Streett constraints of $\mathcal{M}$. The automaton starts in the initial copy, and at some point, guesses the set $F' \subseteq F$ of Streett constraints $(\sigma, \tau)$ such that the region $\tau$ is going to repeat infinitely many times, and switches to the copy corresponding to $F'$. The copy corresponding to the set $F'$ ensures that, for every $(\sigma, \tau) \in F'$, $\tau$ is visited infinitely often, and for every $(\sigma, \tau) \notin F'$, $\sigma$ is not visited. To enforce that, for every $(\sigma, \tau) \in F'$, $\tau$ is visited infinitely often, we introduce a counter as in the translation from multi-Büchi constraints to Büchi constraint. To enforce that, for every $(\sigma, \tau) \notin F'$, $\sigma$ is not visited, we delete the states in $\sigma$. The formal definition of $\mathcal{M}'$ is left as an exercise.

**Remark 12.7** [From Streett to Büchi] If $\mathcal{M}$ is a Streett automaton with $n$ states and $\ell$ Streett constraints, the corresponding language-equivalent Büchi automaton constructed according to the proof of Proposition 12.5, has $n + n \cdot \ell \cdot 2^\ell$ states. Thus, simulating a set of Streett constraints by a single Büchi constraint causes a blow-up exponential in the number of constraints. Such a blow-up can be shown to be essential.

Given a Streett automaton, we can first construct the equivalent Büchi automaton, and then complement it using the complementation construction for Büchi automata.

**Theorem 12.3** [Complementation of Streett automata] Given a Streett automaton $\mathcal{M}$ with $n$ states and $\ell$ Streett constraints, there exists a Büchi automaton $\mathcal{M}'$ with $2^{O(n^3 \cdot 2^\ell)}$ states such that $L(\mathcal{M}') = \mathbb{A}^\infty \backslash L(\mathcal{M})$.

**Exercise 12.6** {T3} [Complementing deterministic Streett automata] Let $\mathcal{M} = (K, F)$ be a Streett automaton such that $K$ is a deterministic and complete observation structure. Show that if $F$ is interpreted as a Rabin accepting condition, then the resulting language is the complement of $L(\mathcal{M})$.

### 12.3 Expressiveness

#### 12.3.1 $\omega$-regular languages

In Chapter 10, we defined different ways of constructing $\omega$-languages from languages of finite words. In particular, we defined the operators $\text{safe}$, $\text{guar}$, $\text{recur}$, and $\text{persist}$. If the languages to which these operators are applied are regular, then the resulting $\omega$-languages are $\omega$-regular.
**ω-REGULAR LANGUAGES**

The ω-language $\mathcal{L} \subseteq A^\omega$ is a regular-safety language if there is a regular language $L \subseteq A^*$ such that $\mathcal{L} = \text{safe}(L)$. The ω-language $\mathcal{L} \subseteq A^\omega$ is a regular-guarantee language if there is a regular language $L \subseteq A^*$ such that $\mathcal{L} = \text{guar}(L)$. The ω-language $\mathcal{L} \subseteq A^\omega$ is a regular-response language if there is a regular language $L \subseteq A^*$ such that $\mathcal{L} = \text{recur}(L)$. The ω-language $\mathcal{L} \subseteq A^\omega$ is a regular-persistence language if there is a regular language $L \subseteq A^*$ such that $\mathcal{L} = \text{persist}(L)$.

The ω-language $\mathcal{L} \subseteq A^\omega$ is ω-regular if it is a boolean combination of regular-response and regular-persistence languages.

**Remark 12.8** [Normal form for ω-regular languages] Every ω-regular language is of the form

$$\bigcap 0 \leq i \leq k \text{ recur}(L_i) \cup \text{ persist}(L'_i)$$

for regular languages $L_i, L'_i$. ■

Thus, the set of regular-safety languages is a subset of the set of safety languages, etc. The set of ω-regular languages is a subset of the set of reactivity languages.

**Example 12.7** [ω-regular languages] Let $A = \{a, b, c\}$. The ω-language $(Aa)^\omega$ is regular-safe; the ω-language $A^*aA^\omega$ is regular-guarantee; the ω-language $(A^*a)^\omega$ is regular-response; the ω-language $A^*a^\omega$ is regular-persistence; and the ω-language consisting of ω-words with infinitely many $b$ symbols or only finitely many $a$ symbols is ω-regular. The ω-language consisting of ω-words $\bar{a}$ such that for all $i \geq 0$, if $i$ is a prime number, then $a_i = a$, is safe but not regular-safe. ■

Closure properties of regular-safety, regular-guarantee, regular-response, regular-persistence, and ω-regular languages coincide with the corresponding closure properties of safety, guarantee, response, persistence, and reactivity languages, respectively. In particular, regular-safety and regular-guarantee classes are duals of each other, and so are regular-response and regular-persistence classes. These properties are summarized in the following proposition, and its proof follows from the constructions of Section 10.1.3.

**Proposition 12.6** [Closure properties of ω-regular languages] Regular-safety languages, regular-guarantee languages, regular-response languages, and regular-persistence languages are closed under union and intersection, but not under complementation. The ω-regular languages are closed under all boolean operations.
12.3.2 Expressiveness of $\omega$-automata

The $\omega$-language $L \subseteq A^\omega$ is said to be a B"uchi language if $L$ is accepted by some B"uchi automaton. First, let us note that B"uchi languages are closed under all boolean operations:

**Proposition 12.7** [Closure of B"uchi languages] The class of B"uchi languages is closed under union, intersection, and complementation.

Since B"uchi acceptance is a special case of Streett acceptance, and by Proposition 12.5 is powerful enough to admit translation from Streett constraints, it follows that

**Corollary 12.2** [Streett acceptance vs. B"uchi acceptance] The $\omega$-language $L \subseteq A^\omega$ is a a B"uchi language iff $L$ is accepted by some Streett automaton.

**Exercise 12.7** {T3} [Acceptance by fair structures] Show that the $\omega$-language $L \subseteq A^\omega$ is a B"uchi language iff $L$ is the fair language of some finite fair structure.

Recall that every $\omega$-regular language is a boolean combination of regular-response languages. Every regular-response language is accepted by a deterministic B"uchi automaton.

**Proposition 12.8** [From regular-response to deterministic B"uchi] For every regular language $L$, there exists a B"uchi automaton that accepts the response language $\text{recur}(L)$.

**Proof.** Let $L$ be a regular language. There exists a deterministic and complete automaton $M = (K, \sigma^A)$ such that $L_M = L$. Consider an $\omega$-word $\overline{a}$. There exists precisely one initialized $\omega$-trajectory $\overline{s}$ of $K$ such that $\langle \overline{s} \rangle = \overline{a}$. For every $i \geq 0$, the prefix $\overline{s}_0 \ldots i$ belongs to $L$ iff $s_i \in \sigma^A$. Hence, the $\omega$-word $\overline{a}$ belongs to $\text{recur}(L)$ iff the $\omega$-trajectory $\overline{a}$ is $\sigma^A$-fair. It follows that if we interpret $M$ as a B"uchi automaton it accepts the $\omega$-language $\text{recur}(L)$.

**Corollary 12.3** [Lower bound on B"uchi expressiveness] For every $\omega$-regular language $L$, there exists a B"uchi automaton that accepts $L$.

**Proof.** Follows from the definition of $\omega$-regular languages, Proposition 12.7, and Proposition 12.8.

Conversely, the language accepted by every $\omega$-automaton is $\omega$-regular:

**Proposition 12.9** [Upper bound on B"uchi expressiveness] Every B"uchi language is $\omega$-regular.
Proof. To be added.

Exercise 12.8 {T4} [Expressiveness of Muller automata] Show that an $\omega$-language $\omega$-regular iff it is accepted by a Muller automaton.

Exercise 12.9 {T4} [$\omega$-regular expressions] We have been using $\omega$-regular expressions (e.g. $(A^*a)^\omega$) to specify $\omega$-languages. Let us now formally define the syntax of $\omega$-regular expressions. Given a finite alphabet $A$, the set of $\omega$-regular expressions is defined by the grammar

$$\varphi ::= a | \varphi \cdot \varphi | \varphi + \varphi | \varphi^* | \varphi^\omega$$

where $a \in A$. Show that an $\omega$-language $L$ is defined by an $\omega$-regular expression iff $L$ is $\omega$-regular.

Exercise 12.10 {T3} [Expressiveness of CoBuchi automata] Does the class of languages accepted by CoBuchi automata coincide with $\omega$-regular languages?

12.3.3 Deterministic $\omega$-automata

We have established that the nondeterministic varieties of different types $\omega$-automata accept the same class of languages, namely, $\omega$-regular languages. We proceed to understand the expressive power of different types of deterministic $\omega$-automata.

The next proposition shows that every $\omega$-regular language is accepted by some deterministic Streett automaton, and both nondeterministic and deterministic varieties of Streett automata have the same expressive power, namely, $\omega$-regular languages.

Proposition 12.10 [Expressiveness of deterministic Streett] For every $\omega$-regular language $L$, there exists a deterministic Streett automaton that accepts $L$.

Proof. Let $L$ be an $\omega$-regular language. Suppose $L = \bigcap_{0 \leq i \leq k} (\text{recur}(L_i) \cup \text{persist}(L_i'))$ such that the languages $L_i, L_i'$ are regular. For $0 \leq i \leq k$, let $M_i = (K_i, \sigma_i)$ be a complete deterministic automaton accepting the regular language $L_i$, and let $M'_i = (K'_i, \sigma'_i)$ be a complete deterministic automaton accepting the regular language $L_i'$. Let $K$ be the product of the $2k + 2$ observation structures $K_i, K_i'$, for $0 \leq i \leq k$. Since product construction preserves determinism, $K$ is deterministic.

Let $a$ be an $\omega$-word. For every $0 \leq i \leq k$, the structure $K_i$ has precisely one $\omega$-trajectory $s_i$ with the corresponding trace $a$, and the word $a \in \text{recur}(L_i)$ iff the $\omega$-trajectory $s_i$ is $\sigma_i$-fair. Similarly, for every $0 \leq i \leq k$, the structure $K_i'$ has precisely one $\omega$-trajectory $s_i'$ with the corresponding trace $a$, and the word
\( a \in \text{persist}(L') \) iff the \( \omega \)-trajectory \( \sigma \) is not \( \sigma' \)-fair. The product structure \( K \) has precisely one \( \omega \)-trajectory \( \sigma \) with the trace \( a \). For all \( 0 \leq i \leq k \), the \( \omega \)-trajectory \( \sigma_i \) of \( K_i \) is \( \sigma \)-fair iff the \( \omega \)-trajectory \( \sigma \) of \( K \) is \( (\sigma_i) \)-fair (recall the definition of the lifting: \( \sigma_i \) contains all product states whose component corresponding to the structure \( K_i \) is in \( \sigma_i \)). Similarly, for all \( 0 \leq i \leq k \), the \( \omega \)-trajectory \( \sigma_i' \) of \( K_i' \) is \( \sigma' \)-fair iff the \( \omega \)-trajectory \( \sigma \) of \( K \) is \( (\sigma_i') \)-fair.

It follows that the word \( a \) belongs to \( L \) iff the \( \omega \)-trajectory \( \sigma \) of the product is, for all \( 0 \leq i \leq k \), either \( (\sigma_i) \)-fair or not \( (\sigma_i') \)-fair. Hence, the deterministic Streett automaton \( (K, \{(\sigma_i), (\sigma_i') \mid 0 \leq i \leq k\}) \) accepts the \( \omega \)-language \( L \).

**Exercise 12.11** \( \text{T3} \) [Expressiveness of deterministic Rabin automata] Show that an \( \omega \)-language is \( \omega \)-regular iff it is accepted by some deterministic Rabin automaton.

It turns out that Büchi accepting condition is not expressive to capture all \( \omega \)-regular languages, if we restrict to deterministic observation structures. The class of languages accepted by deterministic Büchi automata coincides with the regular-response languages.

**Proposition 12.11** [Expressiveness of deterministic Büchi] The \( \omega \)-language \( L \subseteq A^\omega \) is accepted by a deterministic Büchi automaton iff it is a regular-response language.

**Proof.** By Proposition 12.10, we know that every regular-response language is accepted by a deterministic Büchi automaton. For converse, consider a deterministic Büchi automaton \( M = (K, \sigma^A) \). Let \( L \) be the regular language accepted by the automaton with observation strcuture \( K \) and accepting region \( \sigma^A \). Then, the Büchi automaton \( M \) accepts the regular-response language \( \text{recur}(L) \).

It follows that deterministic Büchi automata are not closed under complementation. For instance, the response language “infinitely many \( a \) symbols” is accepted by a deterministic Büchi automaton, but its complement “only finitely
many $a$ symbols” is a persistence language, and is not accepted by any deterministic Büchi automaton. Intuitively, to define the persistence language consisting of $\omega$-words with a suffix containing only $b$ symbols using Büchi acceptance, the automaton must “guess” when the suffix containing only $b$ symbols has commenced. The relationship between different classes of $\omega$-languages is summarized in Figure 12.7

**Exercise 12.12** (T3) [Union closure of deterministic Büchi automata] Since response languages are closed under union, from Proposition 12.11, it follows that deterministic Büchi automata are closed under union. Closure under union can, alternatively, be established by a direct construction. Give an algorithm to construct, given two deterministic Büchi automata $M_1$ and $M_2$, a deterministic Büchi automaton that accepts the union $L_{M_1} \cup L_{M_2}$. ■

**Appendix: Notation**

For two sets $A$ and $B$, if $\sigma$ is a subset of $A$ then $\sigma \uparrow$ denotes the subset $\{(a, b) \mid a \in \sigma\}$ of the product-set $A \times B$; if $\sigma$ is a subset of $B$ then $\sigma \uparrow$ denotes the subset $\{(a, b) \mid b \in \sigma\}$ of the product-set $A \times B$. The lifting operator $\uparrow$ can similarly be applied to binary relations over $A$ to obtain binary relations over the product-set. The lifting operation generalizes to products of multiples sets also.