Theoretical Computer Science 110 (1993) 1–51
Elsevier

Fundamental Study

X-automata on ω-words

Joost Engelfriet and Hendrik Jan Hoogeboom

Department of Computer Science, Leiden University, P.O. Box 9512, 2300 RA Leiden, The Netherlands

Communicated by M. Nivat
Received October 1991

Abstract

Engelfriet, J. and H.J. Hoogeboom, X-automata on ω-words, Theoretical Computer Science 110 (1993) 1–51.

For any storage type X, the ω-languages accepted by X-automata are investigated. Six accepting conditions (including those introduced by Landweber) are compared for X-automata. The inclusions between the corresponding six families of ω-languages are essentially the same as for finite-state automata. Apart from unrestricted automata, also real-time and deterministic automata are considered. The main tools for this investigation are: (1) a characterization of the ω-languages accepted by X-automata in terms of inverse X-transductions of finite-state ω-languages; and (2) the existence of topological upper bounds on some of the families of accepted ω-languages (independent of the storage type X).

Contents

Correspondence to: H.J. Hoogeboom, Department of Computer Science, Leiden University, P.O. Box 9512, 2300 RA Leiden, The Netherlands. Email: hjh@rulwinw.leidenuniv.nl.

Introduction

An automaton $\mathscr{A}$ that is meant to work on finite input words may as well be given an infinite input word u: it works on u as if u were a "very large" finite word. The essential difference is in the way that $\mathscr{A}$ accepts u; obviously, one cannot use acceptance by final state as for finite words.

The first one to use automata to accept infinite words, with a particular acceptance criterion, was Büchi (in solving a decision problem in logic, [3]). Another criterion was given by Muller [26]. A deterministic finite-state automaton $\mathscr{A}$ accepts an infinite word u in the fashion of Muller if the set of states entered by $\mathscr{A}$ infinitely often during its computation on u belongs to a given family of "final" state sets. This *family* replaces the usual *set* of final states. Five criteria for accepting infinite words were proposed by Landweber in [20], including those introduced by Büchi and Muller, and he characterized the five corresponding families of infinitary languages accepted by deterministic finite-state automata in a topological setting.

The relative power of these five acceptance criteria was subsequently compared for (deterministic and nondeterministic) finite-state automata [17, 34] pushdown automata [23, 6, 7], Turing machines [40, 8], and Petri nets [38]. If one compares the results of these investigations, one notices some striking similarities. It seems that the acceptance types have the same relative power independently of the storage used by the automaton involved. Moreover, as for finite-state automata, connections between acceptance types and the lower levels of the topologically defined Borel hierarchy can also be observed for deterministic pushdown automata and Turing machines (see the survey [32]). These observations are the main motivation for the present paper. Using a general framework, we want to explain the similarities between the results obtained for the various specific types of automata (as is done for automata on finite words in [12]). Our abstract model of storage is called a *storage type*. It describes the storage configurations, together with the tests and transformations that can be applied to them. Automata equipped with a specific storage X (and a one-way input tape) are called X-automata. We study six (rather than five) families of ω-languages that can be accepted by an X-automaton using six different acceptance criteria on the sequence of

states entered by the automaton during a computation. (It should be noted that acceptance can also be defined in terms of the storage configurations rather than the states, see [28], but this will give quite different results, cf. [38]). A possible approach to comparing the six acceptance criteria is by giving constructions on automata that show how one acceptance type can be simulated by another. In fact, as observed in [6, 38], it is not too difficult to generalize most of the constructions given in [17] for finite-state automata, simply by "adding" storage instructions to the transitions. Hence, it is not much of a surprise that the inclusions between the six families for X-automata are similar to those formed by the families for finite-state automata. Of course, this is a rather boring and time-consuming approach. Also, if one wants to study X-automata satisfying a particular property (as, e.g., being real-time or deterministic), it is necessary to check each of the constructions to see whether it preserves the property under consideration (and if not, to adapt the construction). We use a more efficient way of transferring the results for finite-state automata to arbitrary storage types. Our main tool is a characterization of the ω-languages accepted by X-automata in terms of (infinitary) transductions applied to the ω-languages accepted by finite-state automata. Since we do not use the acceptance criteria to define transductions, this single result can be used to show that the inclusions that hold between the six families of finite-state ω-languages are also valid for X-automata.

This, of course, does not indicate whether or not an inclusion is strict. We show that the topological upper bounds on the complexity of accepted languages as given by Landweber for deterministic finite-state automata can be generalized to X-automata (as already suggested in [20]). This implies that for deterministic X-automata the inclusions are always strict. The same result holds for real-time automata.

Besides investigating the relative power of the six acceptance criteria, we also study the expressive power of real-time automata and deterministic automata, relative to unrestricted automata.

Section 1 contains the preliminaries to this paper. It introduces our notation on infinite words, and the few topological notions that we will need. In the second section we formalize the notions of storage type, automaton, and transducer, and we define the six different acceptance types we use in accepting infinitary languages. Apart from definitions, Section 2 already contains some preliminary results that are used in the rest of the paper.

In Section 3 we study both nondeterministic and deterministic X-automata. First we present the above-mentioned characterization of the corresponding families of ω-languages (Theorem 3.3). From this, we obtain the hierarchy for ω-languages accepted by nondeterministic X-automata (Theorem 3.5). For specific storage types the hierarchy can be strict (to be more precise, it can contain three different families) or it can collapse into a single family. We give a sufficient condition for such a collapse (Theorem 3.11): the six families of ω-languages accepted by X-automata are all equal when the storage X can simulate an additional (blind) counter. We formalize this notion of simulation of one storage type by another in terms of deterministic transductions.

Real-time automata are investigated in Section 4. The inclusions between the families of ω-languages accepted by real-time automata are very similar to those found in Section 3. Here, however, the inclusions are always strict (Theorem 4.9). The counterexamples are obtained by establishing topological upper bounds that are independent of the storage type. In fact, these results can be extended to the larger class of automata that do not have an infinite computation on a finite input. We call this property *finite delay*. In the final part of Section 4 we compare the expressive power of real-time automata, automata with finite delay, and unrestricted automata. On the one hand, we obtain the result that real-time automata are as powerful as automata with finite delay for any storage type that can simulate an additional queue "in real time" (Theorem 4.17). On the other hand, however, we discuss a storage type for which real-time is less powerful than finite delay (Theorem 4.19). The power of finite-delay automata may be less than or the same as that of unrestricted automata, depending on the storage type (Theorem 4.21).

We return to deterministic automata in Section 5. Again we obtain topological upper bounds on the accepted ω-languages. Together with our basic characterization (given in Section 3) this is used to establish a proper hierarchy similar to the hierarchy for deterministic finite-state automata (Theorem 5.5). The expressive power of deterministic automata vs. nondeterministic automata is also discussed (Theorem 5.6).

In Section 6 we study a storage type of "maximal power", in the sense that it can simulate any other storage type. The families of ω-languages accepted by automata of this type coincide with the topological upper bounds mentioned above, that belong to the lower levels of the topological hierarchy of Borel sets (Theorems 6.3 and 6.6). These results are similar to those obtained in [1, 30] for transition systems. They illustrate once more the strong connections between acceptance type and topological complexity.

In the final section we discuss the possibility of studying arbitrary acceptance criteria (perhaps based on logic) rather than the six to which we have restricted ourselves in the first six sections.

An extended abstract of this paper was presented at ICALP 89 [10].

1. Preliminaries

We assume the reader to be familiar with the basic notions of the theory of infinitary languages, e.g., as discussed in one of the following surveys and introductions: [9, 15, 32, 36]. In this section we fix our notation and terminology, and we recall the topological notions relevant to our paper.

1.1. Sets and functions, infinite words

We use $\mathbb{N}$ to denote the set of nonnegative integers. The symbol $\subseteq$ ($\subset$) denotes set inclusion (strict set inclusion); in diagrams we will use $\Rightarrow$ ($\rightarrow$). We use $\sqcap$ to indicate

that sets intersect, i.e., $X \sqcap Y$ if $X \cap Y \neq \emptyset$. For a family A of sets, $\bigcup A$ denotes the union of all elements of A.

We use the following notations for a relation $R \subseteq X \times Y$.

$R^{-1} = \{(y, x) \in Y \times X \mid (x, y) \in R\}$, for $X' \subseteq X$, $R(X') = \{y \in Y \mid (x, y) \in R$ for some $x \in X'\}$, $ran(R) = R(X)$, and $dom(R) = ran(R^{-1})$. If $\mathscr{R}$ is a family of relations and $\mathscr{A}$ is a family of sets, then $\mathscr{R}^{-1} = \{R^{-1} \mid R \in \mathscr{R}\}$, $\mathscr{R}(\mathscr{A}) = \{R(A) \mid R \in \mathscr{R}, A \in \mathscr{A}\}$, $dom(\mathscr{R}) = \{dom(R) \mid R \in \mathscr{R}\}$, and $ran(\mathscr{R}) = \{ran(R) \mid R \in \mathscr{R}\}$.

A mapping $f: A \to Y$, where $A = \mathbb{N}$ or $A = \{0, 1, \ldots, n-1\}$ for some $n \in \mathbb{N}$, is called a *sequence (over Y)*; it will sometimes be specified in the form $\langle f(i) \rangle_{i \in A}$. f is infinite in case $A = \mathbb{N}$ and finite in case $A = \{0, 1, \ldots, n-1\}$ for some $n \in \mathbb{N}$; in the latter case n is the length of f, denoted by $|f|$.

Let Σ be an alphabet. A sequence over Σ is called a (finite or infinite) *word* over Σ. An infinite word over Σ is also called an *ω-word* over Σ. The set of all finite words over Σ, including the empty word Λ, is denoted by Σ^*, and the set of all ω-words over Σ by Σ^ω. Since a finite or infinite word u over Σ is a mapping $u: A \to \Sigma$, $u(i)$ denotes the $(i+1)$st letter of u (if it exists). A subset of Σ^* is called a *finitary language* (or just *language*) over Σ; an *ω-language* (or *infinitary language*) over Σ is a subset of Σ^ω.

The concatenation of a finite word x and an ω-word u is the ω-word $x.u$ defined by $(x.u)(i) = x(i)$ if $i \leqslant |x|$ and $(x.u)(i) = u(i - |x|)$ otherwise. A finite word x is a prefix of the ω-word v if there exists an ω-word u such that $x.u = v$. For a finite or infinite word v, $v[n]$ denotes the prefix of length n of v (when it exists), and $pref(v)$ denotes the set of (finite) prefixes of v. For a finitary or infinitary language K, $pref(K) = \bigcup \{pref(v) \mid v \in K\}$.

An infinite sequence of finite words $\langle x_i \rangle_{i \in \mathbb{N}}$ such that each x_i is a prefix of its successor x_{i+1} defines a unique element u of $\Sigma^* \cup \Sigma^\omega$ by taking the "least upper bound" of the sequence, i.e., the shortest u that has each x_i as a prefix. u will be denoted by $lub\langle x_i \rangle_{i \in \mathbb{N}}$. Note that u can only be finite if the sequence is eventually stationary, i.e., if there exists a constant N such that $u = x_i$ for $i \geqslant N$.

Definition 1.1. Let $K \subseteq \Sigma^*$ be a finitary language.
The *ω-power* of K, denoted by K^ω, is the ω-language

$$\{u \in \Sigma^\omega \mid u = lub\langle x_i \rangle_{i \in \mathbb{N}}, \text{ where } x_0 \in K \text{ and } x_{i+1} \in x_i.K \text{ for } i \in \mathbb{N}\},$$

the *adherence* of K, denoted by $adh(K)$, is the ω-language

$$\{u \in \Sigma^\omega \mid pref(u) \subseteq pref(K)\},$$

and the *limit* of K, denoted by $lim(K)$, is the ω-language

$$\{u \in \Sigma^\omega \mid pref(u) \cap K \text{ is infinite}\}.$$

1.2. Topology

Σ^ω can be turned into a compact metric space by defining the distance function

$$d(u, v) = \begin{cases} 0 & \text{if } u = v, \\ 2^{-\min\{n \mid u[n] \neq v[n]\}} & \text{if } u \neq v. \end{cases}$$

With this distance, the open sphere of radius 2^{-n} around $u \in \Sigma^\omega$ is the set $u[n].\Sigma^\omega$. The induced topology coincides with the product topology of Σ^ω (with the discrete topology on Σ), and is sometimes called the *natural topology* on Σ^ω.

We will use $\mathcal{G}$ and $\mathcal{F}$ to denote the family of open and closed subsets of Σ^ω, respectively. These families form the basis of a hierarchy known as the *Borel hierarchy*. It consists of the families $\mathcal{G}, \mathcal{G}_\delta, \mathcal{G}_{\delta\sigma}, \ldots$ and the families $\mathcal{F}, \mathcal{F}_\sigma, \mathcal{F}_{\sigma\delta}, \ldots$, where, for a family $\mathcal{X}$, $\mathcal{X}_\delta$ ($\mathcal{X}_\sigma$) is the family of denumerable intersections (denumerable unions) of $\mathcal{X}$-sets. Thus, in particular, $\mathcal{G}_\delta$ is the family of denumerable intersections of open sets, and $\mathcal{F}_\sigma$ is the family of denumerable unions of closed sets. Note that $\mathcal{F} \cup \mathcal{G} \subseteq \mathcal{F}_\sigma \cap \mathcal{G}_\delta$.

There is a close correspondence between the infinitary languages that are in the lower levels of the Borel hierarchy and the language-theoretic operations given above (see, e.g., [20, 22, 34, 2]).

Proposition 1.2. *Let $L \subseteq \Sigma^\omega$. Then*

(1) *$L \in \mathcal{G}$ if and only if $L = K.\Sigma^\omega$ for some $K \subseteq \Sigma^*$.*
(2) *$L \in \mathcal{F}$ if and only if $L = adh(K)$ for some $K \subseteq \Sigma^*$.*
(3) *$L \in \mathcal{G}_\delta$ if and only if $L = lim(K)$ for some $K \subseteq \Sigma^*$.*

The Borel hierarchy is proper at each level, but in this paper we need this fact for the lowest two levels only. Using the above characterizations it is not difficult to give examples of ω-languages that separate the Borel families $\mathcal{F}$ and $\mathcal{G}$, and the families $\mathcal{F}_\sigma$ and $\mathcal{G}_\delta$ (see, e.g., [20, Lemma 3.1]). Recall that $\mathcal{F}_\sigma$ and $\mathcal{G}_\delta$ (like $\mathcal{F}$ and $\mathcal{G}$) are "complementary", in the sense that $L \subseteq \Sigma^\omega$ belongs to $\mathcal{F}_\sigma$ if and only if its complement $\Sigma^\omega - L$ belongs to $\mathcal{G}_\delta$.

Proposition 1.3. (1) $\{0, 1\}.1^\omega \in \mathcal{F} - \mathcal{G}$.
(2) $0^*1.\{0, 1\}^\omega \in \mathcal{G} - \mathcal{F}$.
(3) $0^*.1^\omega \in (\mathcal{F}_\sigma \cap \mathcal{G}_\delta) - (\mathcal{F} \cup \mathcal{G})$.
(4) $\{0, 1\}^*.1^\omega \in \mathcal{F}_\sigma - \mathcal{G}_\delta$.
(5) $(0^*1)^\omega \in \mathcal{G}_\delta - \mathcal{F}_\sigma$.

2. Automata on ω-words

In this section we formalize how we use automata with storage to define ω-languages. In the first subsection we define the notions of storage type and transducer (i.e., automaton with input and output). In Section 2.2. we fix our notation concerning

acceptance of ω-languages (using six different criteria). The first two (technical) properties concerning ω-languages are given in Section 2.3. We consider the various families of ω-languages accepted by (nondeterministic and deterministic) finite-state automata in Section 2.4. In particular, we recall the relations between these classes; they will play an important role in the sequel of the paper as we will use them to obtain similar relations for the families of ω-languages accepted by automata with arbitrary storage. Finally, in Section 2.5, we present some elementary results on transductions.

2.1. Storage and automata

For finite words, the general notion of an automaton, using some kind of storage, was introduced in [16, 27, 13]. The resulting AFA theory (abstract families of automata) provides a useful framework for a uniform investigation of different types of automata (see [12]). Here we attempt to set up a similar theory for automata on infinite words (see also [28]). The basic definitions can be taken over in a straightforward way. The particular variation of AFA theory that we use is similar to the one in [11].

The basic constituents of a storage type are a set of configurations, together with sets of symbols representing tests and transformations that can be applied to these configurations, and an "interpretation" of these symbols.

Definition 2.1. A *storage type* is a 5-tuple $X=(C, C_{in}, P, F, \mu)$, where
- C is a set of (*storage*) *configurations*,
- $C_{in} \subseteq C$ is a set of *initial* (*storage*) *configurations*,
- P is a set of *predicate symbols*,
- F is a set of *instruction symbols*, $P \cap F = \emptyset$, and
- μ is a *meaning function*, which assigns to each $p \in P$ a (total) mapping $\mu(p): C \to \{true, false\}$, and to each $f \in F$ a partial function $\mu(f): C \to C$.

The set of all Boolean expressions over P, using the Boolean connectives $\wedge$, $\vee$ and $\neg$, and the constants *true* and *false*, is denoted by $BE(P)$; elements of this set are called *tests*. The meaning function is extended to $BE(P)$ in the obvious way. We extend μ also from F to F^* by defining $\mu(\Lambda)$ to be the identity on C and by setting $\mu(f\varphi) = \mu(\varphi) \circ \mu(f)$ for $\varphi \in F^*$ and $f \in F$, where $\circ$ denotes function composition.

Example 2.2. The storage type *pushdown*, denoted PD, is defined by $PD=(C, C_{in}, P, F, \mu)$, where

$C = \Gamma^+$, for a fixed infinite set Γ (of pushdown symbols),

$C_{in} = \Gamma$,

$P = \{top = \gamma \mid \gamma \in \Gamma\} \cup \{bottom\}$,

$F = \{push(\gamma) \mid \gamma \in \Gamma\} \cup \{pop\}$,

and, for $c=au$ with $a\in\Gamma$ and $u\in\Gamma^*$,

$$\mu(top=\gamma)(c)=true \text{ iff } \gamma=a,$$

$$\mu(bottom)(c)=true \text{ iff } u=\Lambda,$$

$$\mu(push(\gamma))(c)=\gamma c,$$

$$\mu(pop)(c)=u \text{ if } u\neq\Lambda, \text{ and undefined otherwise.}$$

The storage type *counter* equals $CTR=(\mathbb{N}, \{0\}, \{zero\}, \{incr, decr\}, \mu)$, where for $n\in\mathbb{N}$,

$$\mu(zero)(n)=true \text{ iff } n=0,$$

$$\mu(incr)(n)=n+1, \text{ and}$$

$$\mu(decr)(n)=n-1 \text{ if } n\geqslant 1, \text{ and undefined if } n=0.$$

Definition 2.3. Let $X=(C, C_{in}, P, F, \mu)$ be a storage type. An *X-transducer* is a construct $\mathscr{A}=(Q, \Sigma, \delta, q_{in}, c_{in}, \Delta)$, where
- Q is the finite set of *states*,
- Σ is the *input alphabet*,
- Δ is the *output alphabet*,
- the *finite control* δ is a finite subset of $Q\times(\Sigma\cup\{\Lambda\})\times BE(P)\times Q\times F^*\times\Delta^*$, elements of which are called *transitions*,
- $q_{in}\in Q$ is the *initial state*, and
- $c_{in}\in C_{in}$ is the *initial storage configuration*.

Note that an X-transducer has no final states. These will be treated later (for finite words only).

Let $\mathscr{A}=(Q, \Sigma, \delta, q_{in}, c_{in}, \Delta)$ be an X-transducer for some storage type $X=(C, C_{in}, P, F, \mu)$. A transition $(q, a, \beta, q', \varphi, w)$ is a Λ-*transition* if $a=\Lambda$. $\mathscr{A}$ is *real-time* if it has no Λ-transitions or, equivalently, if δ is a subset of $Q\times\Sigma\times BE(P)\times Q\times F^*\times\Delta^*$.

$\mathscr{A}$ is *deterministic* if, for every two different transitions $(q_i, a_i, \beta_i, q_i', \varphi_i, w_i)$, $i=1, 2$, from δ with $q_1=q_2$, either $a_1\neq a_2$ and $a_1, a_2\neq\Lambda$ or $\mu(\beta_1\wedge\beta_2)(c)=false$ for every $c\in C$.

If $|w|=1$ for each transition $(q, a, \beta, q', \varphi, w)$ of $\mathscr{A}$, then $\mathscr{A}$ is 1-*output*.

An *instantaneous description* (*ID*) *of* $\mathscr{A}$ is an element of $Q\times\Sigma^*\times C\times\Delta^*$. The instantaneous description (q, x, c, y) intuitively means that $\mathscr{A}$ is in state q, has read x from the input tape, has c as its storage configuration, and has written y on its output tape. The *step relation of* $\mathscr{A}$, denoted by $\vdash_{\mathscr{A}}$, is the binary relation on $Q\times\Sigma^*\times C\times\Delta^*$ defined by $(q, x, c, y)\vdash_{\mathscr{A}}(q', x', c', y')$ if there exists a transition $(q, a, \beta, q', \varphi, w)\in\delta$ such that $\mu(\beta)(c)=true$, $c'=\mu(\varphi)(c)$, $x'=xa$, and $y'=yw$. Intuitively, this means that if $\mathscr{A}$ is in state q and has the storage configuration c, it may use the transition $(q, a, \beta, q', \varphi, w)$ provided c satisfies the test β, and then it reads a from its

input tape, changes its state to q', performs φ to the storage configuration, and writes w on its output tape. The reflexive and transitive closure of $\vdash_{\mathscr{A}}$ is denoted by $\vdash_{\mathscr{A}}^*$.

An *infinite run* (or just *run*) *of* $\mathscr{A}$ is an infinite sequence $r=\langle\tau_i\rangle_{i\in\mathbb{N}}$ of *ID*s such that $\tau_0=(q_{\text{in}}, \Lambda, c_{\text{in}}, \Lambda)$, and $\tau_i\vdash_{\mathscr{A}}\tau_{i+1}$ for each $i\in\mathbb{N}$; it is a run *on input* $lub\langle x_i\rangle_{i\in\mathbb{N}}$, *with output* $lub\langle y_i\rangle_{i\in\mathbb{N}}$, where $\tau_i=(q_i, x_i, c_i, y_i)$. The sequence $\langle q_i\rangle_{i\in\mathbb{N}}$ is called the *state sequence of* the run r.

If $\mathscr{A}$ has no run on an infinite input word with a finite output word, then $\mathscr{A}$ is called *ω-preserving*. Note that each 1-output transducer is ω-preserving.

The *infinitary transduction* (or just *transduction*) of $\mathscr{A}$, denoted by $T(\mathscr{A})$, is defined as $\{(u, v)\in\Sigma^\omega\times\Delta^\omega|$ there is a run of $\mathscr{A}$ on input u with output $v\}$.

The family of transductions of X-transducers is denoted by XT. The subfamilies of XT consisting of transductions of ω-preserving and 1-output transducers are denoted by XT_ω and $XT_{1-\text{out}}$, respectively. If we consider only deterministic or real-time transducers, we use the prefixes d- and r-, respectively. Thus, e.g., d-PDT_ω denotes the family of infinitary transductions defined by ω-preserving deterministic pushdown transducers. In the same way we use the prefix dr- for transducers that are both deterministic and real-time.

As usual, if $D\subseteq Q$ is a set of final states, then the *finitary transduction* $T_*(\mathscr{A}, D)$ is the set $\{(x, y)\in\Sigma^*\times\Delta^*|(q_{\text{in}}, \Lambda, c_{\text{in}}, \Lambda)\vdash_{\mathscr{A}}^*(q, x, c, y)$ for some $q\in D$ and $c\in C\}$. We use XT_* to denote the family of finitary transductions of X-transducers; the prefixes d-, r-, and dr- are used as above. Note that we do have an acceptance condition for finitary transductions, as opposed to infinitary transductions.

Example 2.4. Let $\mathscr{A}=(Q, \{a, b, c\}, \delta, q_{\text{in}}, c_{\text{in}}, \{d\})$ be the *PD*-transducer with state set $Q=\{q_1, q_2\}$, initial state $q_{\text{in}}=q_1$, initial storage configuration $c_{\text{in}}=\gamma$, and the following transitions (we assume β and γ to be different pushdown symbols):

$(q_1, a, bottom, q_1, \Lambda, \Lambda)$,

$(q_1, b, true, q_1, push(\beta), \Lambda)$,

$(q_1, c, top=\beta, q_2, pop, \Lambda)$,

$(q_2, c, top=\beta, q_2, pop, \Lambda)$, and

$(q_2, \Lambda, bottom, q_1, \Lambda, d)$.

Then $\mathscr{A}$ is neither real-time nor deterministic (since the last two transitions have tests that are both true for the storage configuration β), but it will be deterministic after replacing the test "top$=\beta$" by "$\neg$ *bottom*".

Note that $\mathscr{A}$ has runs on each input from $(\{a\}\cup K)^\omega\cup(\{a\}\cup K)^* b^\omega$, where $K=\{b^n c^n|n\geq 1\}$. However, $\mathscr{A}$ is not ω-preserving and does not produce infinite output for each of these ω-words. More precisely, $T(\mathscr{A})=(a^*.K)^\omega\times\{d\}^\omega$. Changing $\mathscr{A}$ such that the first and second transition have output d yields an ω-preserving

transducer. This will also change the transduction of $\mathscr{A}$ to $(\{a\}\cup K)^\omega \times \{d\}^\omega \cup (\{a\}\cup K)^* b^\omega \times \{d\}^\omega$.

Obviously, the step relation of a transducer $\mathscr{A}$ is not changed by replacing a test β in a transition by an equivalent test, i.e., by a test β' such that $\mu(\beta)=\mu(\beta')$. Neither is it changed by omitting those transitions that have a test which is always false.

Hence, if X is a *blind* storage type (i.e., X has no predicate symbols, cf. [14]), then we may assume that the transitions of an X-transducer are of the form $(q, a, true, q', \varphi, w)$.

A special blind storage type is used to model finite-state transducers; it has neither predicate nor instruction symbols. The *trivial storage type FS* equals $(\{c0\}, \{c0\}, \emptyset, \emptyset, \emptyset)$ for some arbitrary object $c0$. Note that $\emptyset^*=\{\Lambda\}$. Hence, the transitions of an FS-transducer can be assumed to be of the form $(q, a, true, q', \Lambda, w)$.

Finally, we need the notion of the product of two storage types. It combines the power of two storages that can be used in an independent fashion.

Let $X_i=(C_i, C_{\text{in},i}, P_i, F_i, \mu_i)$, $i=1, 2$, be two storage types with $P_1\cap P_2=\emptyset$ and $F_1\cap F_2=\emptyset$. The *product of* X_1 *and* X_2, denoted $X_1\times X_2$, is the storage type $(C, C_{\text{in}}, P, F, \mu)$ with $C=C_1\times C_2$, $C_{\text{in}}=C_{\text{in},1}\times C_{\text{in},2}$, $P=P_1\cup P_2$, $F=F_1\cup F_2$, and μ defined by

$$\mu(p)(c_1, c_2)=\begin{cases}\mu_1(p)(c_1) & \text{if } p\in P_1,\\ \mu_2(p)(c_2) & \text{if } p\in P_2,\end{cases}$$

$$\mu(f)(c_1, c_2)=\begin{cases}(\mu_1(f)(c_1), c_2) & \text{if } f\in F_1,\\ (c_1, \mu_2(f)(c_2)) & \text{if } f\in F_2.\end{cases}$$

It is, of course, also possible to define the product of two storage types that have predicate or instruction symbols in common. In that case we distinguish between these symbols by first renaming them, e.g., by adding a subscript for each of the components.

In a similar way, the product of more than two storage types can be defined. The product of n, $n\geqslant 1$, storage types, all equal to X, is denoted by X^n; we write p_i and f_i to denote the predicate symbol p and the instruction symbol f when applied to the ith component of X^n. It is convenient to define $X^0=FS$.

As an example, the storage $CTR^2=CTR\times CTR$ has two counters that may be incremented, decremented, and tested for zero independently from each other. The instruction $incr_2decr_1$ first increments the second counter and then decrements the first counter (when defined).

An X^*-transducer is an X^n-transducer for some $n\in\mathbb{N}$. We will use X^* as if it were a storage type (indeed, it can be formally defined as such, cf. [12, Lemma 4.5.1]). So, we write, e.g., X^*T to denote $\bigcup_{n\in\mathbb{N}} X^nT$.

In the remainder of this paper $X=(C, C_{\text{in}}, P, F, \mu)$ *denotes an arbitrary storage type.*

2.2. *(σ, ρ)-accepting infinite runs*

We will now discuss how an X-transducer $\mathscr{A}$ may be used to accept ω-languages by imposing acceptance conditions on the state sequences of its runs. Since, in this case, we are not interested in $\mathscr{A}$'s output, $\mathscr{A}$ is called an *X-automaton.* In our notation, we drop the output component from $\mathscr{A}$, and from its transitions and *ID*s.

Let Q be a finite set (of "states") and let $f: \mathbb{N} \to Q$ be a mapping (i.e., an infinite word $f \in Q^\omega$). As for all relations the *range of* f, denoted by $ran(f)$, is the set $\{q \in Q \mid f(i) = q$ for some $i \in \mathbb{N}\}$; the *infinity set of* f, denoted by $inf(f)$, is the set $\{q \in Q \mid f(i) = q$ for infinitely many $i \in \mathbb{N}\}$. Note that $inf(f)$ is nonempty, due to the finiteness of Q; in fact, there exists an $N \in \mathbb{N}$ such that $f(i) \in inf(f)$ for $i \geqslant N$.

Let $\mathscr{D} \subseteq 2^Q$ be a family of subsets of Q. Let ρ be a binary relation over 2^Q and let $\sigma: Q^\omega \to 2^Q$ be a mapping that assigns to each infinite sequence over Q a subset of Q. We say that an infinite sequence $f: \mathbb{N} \to Q$ is *(σ, ρ)-accepting with respect to $\mathscr{D}$* if there exists a set $D \in \mathscr{D}$ such that $\sigma(f)\, \rho\, D$.

In the sequel of this paper we assume that ρ ranges over the relations $\sqcap$, $\subseteq$, or $=$, and that σ is one of the mappings ran or inf. Thus, we consider six types of acceptance. Definitions and results that involve the letters σ and ρ are always assumed to be universally quantified.

The relation between the notation we use (see [34]) and the five types of "i-acceptance" as originally defined in [20] are given in Table 1, together with a short intuitive name for some of these types of acceptance. Recall that $(inf, \sqcap)$ is the acceptance type introduced by Büchi [3], whereas $(inf, =)$-acceptance was first considered by Muller [26] (for deterministic automata). $(ran, =)$-acceptance, not considered by Landweber, was first studied in [34].

More precisely, for a given $\mathscr{D} \subseteq 2^Q$, an infinite sequence f of states is $(ran, \sqcap)$-accepting if at least one state from $\bigcup \mathscr{D}$ occurs in f. It is $(ran, \subseteq)$-accepting if all its states are in D, for some $D \in \mathscr{D}$. It is $(inf, \sqcap)$-accepting if at least one state from $\bigcup \mathscr{D}$ occurs infinitely often in f. It is $(inf, \subseteq)$-accepting if there exist $D \in \mathscr{D}$ and $N \in \mathbb{N}$ such that $f(i) \in D$ for $i \geqslant N$, i.e., all states are in D from some moment onwards (recall that Q is finite). Finally, f is $(ran, =)$-accepting or $(inf, =)$-accepting if $\mathrm{ran}(f) \in \mathscr{D}$ or $inf(f) \in \mathscr{D}$, respectively.

Let $\mathscr{A} = (Q, \Sigma, \delta, q_{in}, c_{in})$ be an X-automaton, and let $\mathscr{D} \subseteq 2^Q$ be a family of subsets of Q. A run of $\mathscr{A}$ is called *(σ, ρ)-accepting with respect to $\mathscr{D}$* if its state sequence is (σ, ρ)-accepting with respect to $\mathscr{D}$.

Table 1

$(ran, \sqcap)$	1-accepting	at least once	
$(ran, \subseteq)$	1′-accepting	always	
$(inf, \sqcap)$	2-accepting	infinitely often	(Büchi)
$(inf, \subseteq)$	2′-accepting	from some moment on	
$(inf, =)$	3-accepting		(Muller)

Definition 2.5. The *ω-language (σ, ρ)-accepted by $\mathscr{A}$ with respect to $\mathscr{D}$*, denoted by $L_{\sigma,\rho}(\mathscr{A}, \mathscr{D})$, is the set $\{u \in \Sigma^\omega |$ there is a run of $\mathscr{A}$ on u that is (σ, ρ)-accepting with respect to $\mathscr{D}\}$.

The family of ω-languages (σ, ρ)-accepted by X-automata (with respect to some family of state sets) is denoted by $XL_{\sigma,\rho}$. As before, the corresponding families of ω-languages (σ, ρ)-accepted by deterministic and/or real-time X-automata are denoted by d-$XL_{\sigma,\rho}$, r-$XL_{\sigma,\rho}$, and dr-$XL_{\sigma,\rho}$.

As usual, for a set D of states of $\mathscr{A}$, the (finitary) *language $L_*(\mathscr{A}, D)$ accepted by $\mathscr{A}$ with respect to D* is the set $\{x \in \Sigma^* | (q_{\text{in}}, \Lambda, c_{\text{in}}) \vdash_{\mathscr{A}}^* (q, x, c)$ for some $q \in D$ and $c \in C\}$.

We wish to stress that we consider acceptance with respect to states rather than acceptance with respect to storage configurations (as in [28]). It was shown in [38] that (for Petri nets) these two approaches give quite different results. This will also be the case in the more general setting of X-automata.

In the literature several other definitions are also used; clearly, in a uniform approach such as the present one, we had to made some choices. It is sometimes required that an X-automaton is "total", e.g., in a "global" sense, meaning that the automaton has a run on every input ω-word, or in a "local" sense, in which there should be an applicable transition for each instantaneous description of the automaton. Requiring totality changes (in general) the families $XL_{\sigma,\rho}$. In our opinion, totality should not be required by definition, but should be treated like any other property such as determinism or real-time. To keep this paper of reasonable length, we decided not to investigate totality. In fact, totality is not as straightforward to define for X-automata in general, mainly due to the presence of Λ-steps, and to the fact that some strorage instructions may be partial functions.

We would also like to stress that in our model an input word can only be accepted using an infinite computation that reads every letter of the input. This differs from the acceptance criterion that is used in some of the work of Staiger and Wagner (e.g., [40]). They require only the existence of an infinite run (satisfying the acceptance condition) reading either the input or a finite prefix of it. As explained in [32, p. 422] this leads to incomparable results, e.g., for Turing machines as obtained in [40] on the one hand, and in [8] on the other. Note that for real-time automata both definitions coincide: an infinite run of a real-time automaton reads every input letter.

Example 2.6. (1) Let $\mathscr{A}$ be the (deterministic and real-time) FS-automaton with state set $Q = \{q_0, q_1\}$, input alphabet $\Sigma = \{0, 1\}$, initial state q_0, and transitions $(q_i, j, \mathit{true}, q_j, \Lambda)$ for $i, j \in \{0, 1\}$. Let $\mathscr{D} = \{\{q_1\}\}$, and $\mathscr{Q} = \{Q\}$. Then

$$L_*(\mathscr{A}, \{q_1\}) = \{0, 1\}^*.1, \qquad L_*(\mathscr{A}, Q) = \{0, 1\}^*,$$

$$L_{ran,\sqcap}(\mathscr{A}, \mathscr{D}) = 0^*1.\{0, 1\}^\omega, \qquad L_{ran,\sqcap}(\mathscr{A}, \mathscr{Q}) = L_{ran,\subseteq}(\mathscr{A}, \mathscr{Q}) = \{0, 1\}^\omega,$$

$$L_{ran,\subseteq}(\mathscr{A}, \mathscr{D}) = L_{ran,=}(\mathscr{A}, \mathscr{D}) = \emptyset, \qquad L_{ran,=}(\mathscr{A}, \mathscr{Q}) = 0^*1.\{0, 1\}^\omega,$$

$$L_{inf,\sqcap}(\mathscr{A}, \mathscr{D}) = (0^* 1)^\omega, \qquad L_{inf,\sqcap}(\mathscr{A}, \mathscr{2}) = L_{inf,\subseteq}(\mathscr{A}, \mathscr{2}) = \{0, 1\}^\omega,$$

$$L_{inf,\subseteq}(\mathscr{A}, \mathscr{D}) = L_{inf,=}(\mathscr{A}, \mathscr{D}) = \{0, 1\}^*.1^\omega, \qquad L_{inf,=}(\mathscr{A}, \mathscr{2}) = (0^*1\ 1^*0)^\omega.$$

(2) Let $\mathscr{A}' = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, c0)$ be the (deterministic and real-time) *FS*-automaton with transitions $(q_i, 0, \mathit{true}, q_i, \Lambda)$, $(q_i, 1, \mathit{true}, q_1, \Lambda)$ for $i \in \{0, 1\}$, and let $\mathscr{D} = \{\{q_1\}\}$. Then $L_{inf,\subseteq}(\mathscr{A}', \mathscr{D}) = 0^*1.\{0, 1\}^\omega$.

The relations between the families (d-)$FSL_{\sigma,\rho}$ will be given in Proposition 2.10.

2.3. *Basic properties*

In this section we give two results that are used at several places in this paper. The first lemma is a reformulation into our framework of the well-known fact that for some of the acceptance types it suffices to consider acceptance with respect to families containing just one state set. It was shown for finite-state automata in [34], and in a more general formulation in [7, Lemma 4.1.2]. For completeness we provide a proof.

Lemma 2.7. *Let $\rho \in \{\subseteq, \sqcap\}$. For every (deterministic) X-automaton $\mathscr{A}$ and family of state sets $\mathscr{D}$ for $\mathscr{A}$ there exist a (deterministic) X-automaton $\mathscr{A}'$ and state set D for $\mathscr{A}'$ such that $L_{\sigma,\rho}(\mathscr{A}, \mathscr{D}) = L_{\sigma,\rho}(\mathscr{A}', \{D\})$.*

Proof. If ρ equals $\sqcap$ the lemma is obvious since $L_{\sigma,\sqcap}(\mathscr{A}, \mathscr{D}) = L_{\sigma,\sqcap}(\mathscr{A}, \{\bigcup \mathscr{D}\})$.

For $\subseteq$ we add to the states of the automaton $\mathscr{A}$, for each state set D of $\mathscr{D}$, a Boolean variable which indicates whether the run has remained within D since a particular moment of time; $\mathscr{A}$ resets this Boolean vector each time the run has been outside each state set from $\mathscr{D}$ since the last reset (or since the start of the run). By definition, a run (of the original automaton) is $(\mathit{ran}, \subseteq)$-accepting if there is at least one state set from which the run never leaves, i.e., if the Boolean vector (of the new automaton) is never reset during the run. Similarly, a run is $(\mathit{inf}, \subseteq)$-accepting if there is a state set from which the run leaves only finitely many times, i.e., if the Boolean vector is reset only finitely many times.

Let $\mathscr{A} = (Q, \Sigma, \delta, q_{\text{in}}, c_{\text{in}})$ and let $\mathscr{D} = \{D_1, \ldots, D_n\}$, $n \geqslant 2$. Then $\mathscr{A}'$ is formally defined as $(Q', \Sigma, \delta', q'_{\text{in}}, c_{\text{in}})$, where $Q' = Q \times \{0, 1\}^n$, $q'_{\text{in}} = (q_{\text{in}}, 1^n)$; for each $(q, a, \beta, q_1, \varphi) \in \delta$ and each $d \in \{0, 1\}^n$, δ' contains the transition $((q, d), a, \beta, (q_1, d_1), \varphi)$, where $d_1 = 1^n$ if $d = 0^n$, and if $d \neq 0^n$, then for $1 \leqslant j \leqslant n$, $d_1(j) = 1$ iff $d(j) = 1$ and $q \in D_j$.

Let $E = \{0, 1\}^n - \{0^n\}$. We then have $L_{\sigma,\subseteq}(\mathscr{A}, \mathscr{D}) = L_{\sigma,\subseteq}(\mathscr{A}', \{Q \times E\})$. We leave the formal details to the reader. □

In several proofs in this paper, given an automaton $\mathscr{A}$ and a family of state sets $\mathscr{D}$ for $\mathscr{A}$, a new automaton $\mathscr{A}'$ (satisfying some particular property) will be

constructed in such a way that there is a clear correspondence between the runs of $\mathscr{A}$ and those of $\mathscr{A}'$. As an example (see Lemma 2.9), it is possible to construct for each FS- automaton $\mathscr{A}$ an equivalent real-time FS-automaton $\mathscr{A}'$, and to give a mapping that relates the set of states entered (infinitely often) during a run of $\mathscr{A}'$ on an ω-word to the set of states entered (infinitely often) during the "original" run of $\mathscr{A}$ on that word, by taking into account the "shortcuts" consisting of Λ-transitions. In such cases it will be possible to specify a family $\mathscr{D}'$ of state sets for $\mathscr{A}'$ such that $\mathscr{A}$ and $\mathscr{A}'$ accept related infinitary languages. Rather than giving this family explicitly in each separate construction, we provide a more general result.

Unfortunately, the statement of the lemma is rather technical – though its proof is elementary. We suggest, therefore, that the lemma is skipped on first reading. The reader may consider the result as a technical justification for an argumentation that (in most cases) is intuitively clear.

Lemma 2.8. *Let $\mathscr{A}$ and $\mathscr{A}'$ be automata with input alphabets Σ and Σ' and state sets Q and Q', respectively. Let $R \subseteq \Sigma^\omega \times (\Sigma')^\omega$, and let $\psi: 2^{Q'} \to 2^Q$ be a mapping that satisfies*

(1) $\psi(A \cup B) = \psi(A) \cup \psi(B)$, *and*

(2) *for each $v \in (\Sigma')^\omega$*

$$\bigcup_{(u,v)\in R} \{\sigma(r) \mid r \text{ is a state sequence of a run of } \mathscr{A} \text{ on } u\}$$
$$= \{\psi(\sigma(r')) \mid r' \text{ is a state sequence of a run of } \mathscr{A}' \text{ on } v\}.$$

Then for each $\mathscr{D} \subseteq 2^Q$ there exists $\mathscr{D}'_\rho \subseteq 2^{Q'}$ such that

$$R(L_{\sigma,\rho}(\mathscr{A}, \mathscr{D})) = L_{\sigma,\rho}(\mathscr{A}', \mathscr{D}'_\rho).$$

Proof. By (2), it suffices to define $\mathscr{D}'_\rho$ in such a way that, for every set $S' \subseteq Q'$, $S' \rho D'$ for some $D \in \mathscr{D}'_\rho$ iff $\psi(S') \rho D$ for some $D \in \mathscr{D}$.

(a) Let $\rho \in \{\subseteq, =\}$. Then the relation ρ is transitive, reflexive, and invariant under ψ (i.e., if $A \rho B$ then $\psi(A) \rho \psi(B)$). These properties allow one to prove that, for arbitrary sets $S' \subseteq Q'$ and $D \subseteq Q$, $\psi(S') \rho D$ if and only if $S' \rho D'$ for some D' with $\psi(D') \rho D$. Now $\mathscr{D}'_\rho = \{D' \subseteq Q' \mid \psi(D') \rho D \text{ for some } D \in \mathscr{D}\}$ satisfies the requirement of the lemma.

(b) We now consider the case that $\rho = \sqcap$. According to (1) $\psi(A) = \bigcup_{a \in A} \psi(\{a\})$. Hence, for arbitrary sets $S' \subseteq Q'$ and $D \subseteq Q$, $\psi(S') \sqcap D$ if and only if $\psi(\{s'\}) \sqcap D$ for some $s' \in S'$ if and only if $S' \sqcap \{q' \in Q' \mid \psi(\{q'\}) \sqcap D\}$. Consequently, $\mathscr{D}'_\sqcap = \{\{q' \in Q' \mid \psi(\{q'\}) \sqcap D\} \mid D \in \mathscr{D}\}$ satisfies the requirement of the lemma.

Note that by a "dual" argument the family $\mathscr{D}'_\subseteq = \{\{q' \in Q' \mid \psi(\{q'\}) \subseteq D\} \mid D \in \mathscr{D}\}$ could have been chosen for the relation $\subseteq$. □

2.4. Finite-state automata

We consider the well-known "hierarchy" of the families (d-)$FSL_{\sigma,\rho}$. It is customary in the literature to define finite-state ω-languages using real-time finite-state

automata. We first show that this restriction does not influence the families of (σ, ρ)-accepted finite-state ω-languages.

Lemma 2.9. (1) $FSL_{\sigma,\rho} = \text{r-}FSL_{\sigma,\rho}$.
(2) $\text{d-}FSL_{\sigma,\rho} = \text{dr-}FSL_{\sigma,\rho}$.

Proof. Let $\mathscr{A} = (Q, \Sigma, \delta, q_{\text{in}}, c0)$ be a finite-state automaton with a family $\mathscr{D}$ of state sets. As for the corresponding result for finitary languages, we may construct an equivalent real-time finite-state automaton by contracting each non-Λ transition with all possible sequences of Λ-transitions. However, since in the infinitary case the intermediate states are considered in the acceptance criteria, we should keep track of these in our new automaton as well.

Construct the real-time finite-state automaton $\mathscr{A}' = (Q', \Sigma, \delta', q'_{\text{in}}, c0)$ as follows: $Q' = Q \times 2^Q$, $q'_{\text{in}} = (q_{\text{in}}, \{q_{\text{in}}\})$, and, for $q, q' \in Q$, $U, U' \subseteq Q$ and $a \in \Sigma$, δ' contains the transition $((q, U), a, \textit{true}, (q', U'), \Lambda)$ if and only if there exists a sequence $q_0, q_1, \ldots, q_k$ $(k \geqslant 0)$ of states in Q such that $U' = \{q_1, \ldots, q_k, q'\}$, $q = q_0$, $(q_{i-1}, \Lambda, \textit{true}, q_i, \Lambda) \in \delta$ for $1 \leqslant i \leqslant k$, and $(q_k, a, \textit{true}, q', \Lambda) \in \delta$.

Our original automaton $\mathscr{A}$ may have Λ-cycles – resulting in arbitrary long runs of the form $(q, \Lambda, c0) \vdash^* (q, \Lambda, c0)$ – so, for one run of $\mathscr{A}'$ on an ω-word u there might be (infinitely) many corresponding runs of $\mathscr{A}$ on u. The set of states occurring (infinitely often) during any of these runs of $\mathscr{A}$ is always equal to the union of the sets that occur (infinitely often) on the second component of the states of the run of $\mathscr{A}'$. Hence, we may use Lemma 2.8 to find a suitable family of state sets $\mathscr{D}'_\rho$ such that $L_{\sigma,\rho}(\mathscr{A}, \mathscr{D}) = L_{\sigma,\rho}(\mathscr{A}', \mathscr{D}'_\rho)$ by choosing R to be the identity on Σ^ω, and ψ to be the mapping that satisfies $\psi(\{(q, U)\}) = U$ [and (1) of Lemma 2.8]. □

Hence, we have the following well-known relationships for the families (d-)$FSL_{\sigma,\rho}$ (see, e.g., [39, Theorem 4] for references). Recall that we use $\rightarrow$ to denote strict set inclusion ($\subset$). The diagram in Fig. 1 is "complete", i.e., families not related in the diagram are incomparable.

Proposition 2.10. *The diagram in Fig. 1 holds. For any two families X and Y from the diagram we have $X \subseteq Y$ if and only if there is a path from X to Y.*

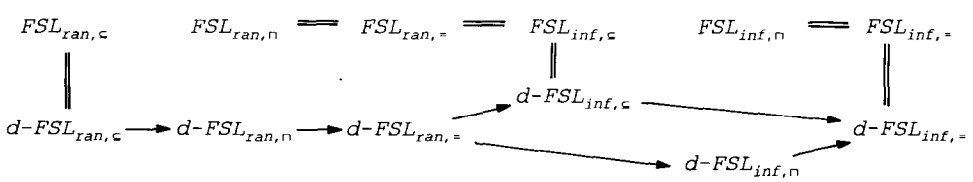

Fig. 1. Inclusion diagram for the finite-state storage type.

Note that when requiring totality (in the "local" sense, i.e., in each state there should be a transition for every letter of the input alphabet) the diagram is slightly different, caused by the fact that $FSL_{ran,\sqcap}$ and d-$FSL_{ran,\sqcap}$ become smaller. Using t- to denote the property of totality in the same way as we use d- and r-, one has t-$FSL_{ran,\sqcap}$ = td-$FSL_{ran,\sqcap} \subset$ d-$FSL_{ran,\sqcap}$. In all other cases t-$FSL_{\sigma,\rho} = FSL_{\sigma,\rho}$ and td-$FSL_{\sigma,\rho}$ = d-$FSL_{\sigma,\rho}$. The families t-$FSL_{ran,\sqcap}$ and $FSL_{ran,\subseteq}$ are incomparable.

An ω-language is called *regular* if it is of the form $\bigcup_{i=1}^{n} K_i.L_i^{\omega}$, where K_i and L_i are regular (finitary) languages. It was proved by McNaughton [25] (see also [9]) that the family of regular ω-languages (sometimes called the ω-regular languages) coincides with d-$FSL_{inf,=}$ ("acceptance by Muller automata") and, consequently, with $FSL_{inf,\sqcap}$ ("acceptance by Büchi automata"). There are several other characterizations of the classes (d-)$FSL_{\sigma,\rho}$ in terms of regular languages and the operations *adh*, *lim*, and ω-power. For example, $FSL_{ran,\subseteq}$ equals the family of adherences of (finitary) regular languages, and $FSL_{inf,\subseteq}$ equals the family of ω-languages of the form $\bigcup_{i=1}^{n} K_i.adh(L_i)$, where K_i and L_i are regular languages. For more details see again, e.g., [39].

2.5. *Transductions*

Infinitary transductions (and, in particular, inverses of ω-preserving transductions) play a fundamental role in this paper: they are used to transfer the relations that hold between the families $FSL_{\sigma,\rho}$ and d-$FSL_{\sigma,\rho}$ to arbitrary storage types. Here we give some preliminary results on families of transductions, mainly concerning their inverses and their domains.

In the finitary case homomorphisms form a subclass of the finite-state transductions. They can be extended to ω-words in the natural way: a (an infinitary) *homomorphism* $h: \Sigma \to \Delta^*$ determines a single-state (real-time and deterministic) FS-transducer that has a transition $(q, a, true, q, h(a))$ for every $a \in \Sigma$. Note that, in general, the resulting transduction is a partial function from Σ^{ω} to Δ^{ω}. We use HOM to denote the family of transductions of such FS-transducers. $HOM_{1\text{-out}}$ denotes the family of corresponding 1-output transductions, obviously corresponding to length-preserving homomorphisms. Although not every homomorphism is ω-preserving (cf. Remark 2.12), we wish to note that every inverse homomorphism is: $HOM^{-1} \subseteq$ dr-$FST^{-1} \subseteq FST_{\omega}$. This is a consequence of the following general result. Recall that $XT^{-1} = \{R^{-1} \mid R \in XT\}$; the notation r-$XT_{1\text{-out}}^{-1}$ abbreviates (r-$XT_{1\text{-out}}$)$^{-1}$.

Lemma 2.11. (1) $XT^{-1} = XT$.

(2) $XT_{1\text{-out}} \subseteq$ r-$XT^{-1} \subseteq XT_{\omega}$, r-$XT_{1\text{-out}}^{-1}$ = r-$XT_{1\text{-out}}$.

Proof. Let $\mathcal{M}$ be an X-transducer with output alphabet Δ. We obtain the inverse of the transduction of $\mathcal{M}$ by replacing each transition $(q, a, \beta, q', \varphi, w)$ by the transition $(q, \Lambda, \beta, q', \varphi, a)$ if $w = \Lambda$, and by the transitions $(q_{i-1}, b_i, true, q_i, \Lambda, \Lambda)$, for $1 \leqslant i < n$, and $(q_{n-1}, b_n, \beta, q', \varphi, a)$, if $w = b_1 \ldots b_n$, $b_i \in \Delta$ for $1 \leqslant i \leqslant n$, where $q_0 = q$ and q_i is a new state

for each $1 \leqslant i < n$. Note that this construction transforms 1-output transducers into real-time transducers. Moreover, it transforms real-time transducers into ω-preserving transducers: since a real-time transducer cannot transform a finite input word into an infinite output word, its inverse (as constructed above) cannot transform an infinite input word into a finite output word. □

Remark 2.12. Although not immediately clear, the families XT and XT_ω are not always equal. As an example consider the (real-time) FS-transducer $\mathcal{M}$ with input alphabet $\{0, 1\}$ and output alphabet $\{1\}$ that deletes all letters 0 from the input. This transducer produces infinite output only if the input contains infinitely many occurrences of the letter 1. Thus, $T(\mathcal{M}) = (0^* 1)^\omega \times \{1\}^\omega$. From the following result we learn that this transduction cannot be realized by an ω-preserving FS-transducer since its domain $(0^* 1)^\omega$ does not belong to $FSL_{ran, \subseteq}$ (see [20]; it also follows from our Corollary 4.8).

The same example shows us that XT_ω is not always closed under inverse. In fact, $T(\mathcal{M}) \notin FST_\omega$, whereas $T(\mathcal{M})^{-1} \in HOM^{-1} \subseteq FST_\omega$.

It is not difficult to show that domains of X-transducers (and of ω-preserving X-transducers) coincide with one of the families of ω-languages accepted by X-automata. Recall that for an X-transducer $\mathcal{M}$ with input alphabet Σ, $dom(T(\mathcal{M})) = \{u \in \Sigma^\omega \mid$ there is a run of $\mathcal{M}$ on u wih infinite output$\}$. For an ω-preserving transducer $\mathcal{M}$ we even have $dom(T(\mathcal{M})) = \{u \in \Sigma^\omega \mid$there is a run of $\mathcal{M}$ on $u\}$.

Lemma 2.13. (1) $dom(XT) = XL_{inf, \sqcap} = ran(XT)$.

(2) $dom(XT_\omega) = XL_{ran, \subseteq}$.

Proof. (1): If we extend the state set Q of the X-transducer $\mathcal{M}$ to the set $Q \times \{0, 1\}$, and replace each transition $(q, a, \beta, q', \varphi, w)$ by the pair of transitions $((q, i), a, \beta, (q', 0), \varphi)$ if $w = \Lambda$ $(i \in \{0, 1\})$ and the pair $((q, i), a, \beta, (q', 1), \varphi)$ if $w \neq \Lambda$ $(i \in \{0, 1\})$, then we obtain an X-automaton $\mathcal{A}$ such that clearly $dom(T(\mathcal{M})) = L_{inf, \sqcap}(\mathcal{A}, \{Q \times \{1\}\})$.

On the other hand, given an X-automaton $\mathcal{A}$ and a family of state sets $\mathcal{D}$ for $\mathcal{A}$, we make it into an X-transducer $\mathcal{M}$ by writing the letter 1 as output for each transition that enters a state from $\bigcup \mathcal{D}$. $\mathcal{M}$ writes Λ when it does not enter a state from $\bigcup \mathcal{D}$. Obviously, $\mathcal{A}$ enters a state from $\bigcup \mathcal{D}$ infinitely often if and only if in its corresponding run $\mathcal{M}$ outputs an infinite word. Hence, $dom(T(\mathcal{M})) = L_{inf, \sqcap}(\mathcal{A}, \mathcal{D})$.

The equality $dom(XT) = ran(XT)$ follows from Lemma 2.11(1).

(2): Let $\mathcal{M}$ be an ω-preserving X-transducer with state set Q. Transform $\mathcal{M}$ into an X-automaton $\mathcal{A}$ by replacing each transition $(q, a, \beta, q', \varphi, w)$ by the transition $(q, a, \beta, q', \varphi)$. Now $dom(T(\mathcal{M})) = \{u \in \Sigma^\omega \mid$ there is a run of $\mathcal{M}$ on $u\} = \{u \in \Sigma^\omega \mid$ there is a run of $\mathcal{A}$ on $u\} = L_{ran, \subseteq}(\mathcal{A}, \{Q\})$.

In order to prove the converse inclusion $dom(XT_\omega) \supseteq XL_{ran, \subseteq}$, consider an X-automaton $\mathcal{A}$ and a family of state sets $\mathcal{D}$ for $\mathcal{A}$. By Lemma 2.7, we may assume that $\mathcal{D}$ consists of a single state set D. Moreover, since no $(ran, \subseteq)$-accepting run enters any

state outside D, we may assume that D equals the set Q of all states of $\mathscr{A}$. By replacing each transition $(q, a, \beta, q', \varphi)$ of $\mathscr{A}$ by the transition $(q, a, \beta, q', \varphi, 1)$ one obtains an (1-output) X-transducer $\mathscr{M}$ with output alphabet $\{1\}$. Clearly, $dom(T(\mathscr{M}))= L_{ran,\subseteq}(\mathscr{A}, \{Q\})$. □

3. The basic characterization

In this section we derive our main characterization result for the families of ω-languages (σ, ρ)-accepted by both nondeterministic and deterministic X-automata. We express these families in terms of (inverse ω-preserving) X-transductions of (σ, ρ)-accepted finite-state ω-languages (Theorem 3.3). This generalizes a result from [40], where families of ω-languages accepted by Turing machines are characterized in the arithmetical hierarchy using transductions by Turing machines as a main tool. Using our characterization we transfer the inclusions (and equalities) that hold for the families $FSL_{\sigma,\rho}$ to the families $XL_{\sigma,\rho}$ for arbitrary storage type X (Theorem 3.5). The corresponding results for real-time and deterministic automata are given in Sections 4 and 5, respectively. This method does not give information concerning the strictness of the inclusions (except, of course, where we have equality in the finite-state case). In fact, it is known from the literature that the two remaining inclusions are strict inclusions for some storage types (such as FS), while they are equalities for others. In Section 3.3 we give a sufficient condition on the storage type to ensure equality for the six families of ω-languages accepted using this storage type (Theorem 3.11). To this aim, we introduce the notion of simulation of one storage type by another, which enables us to compare the strength of storage types.

3.1. Decomposition and composition

In Lemma 3.1 we show how to decompose the work of an X-automaton into two phases: a phase in which the input is processed, and an acceptance phase. The first phase can be realized by an X-transducer (without acceptance criterion), and the second phase by a finite-state automaton (with the same acceptance criterion as the X-automaton).

Lemma 3.1. (1) $XL_{\sigma,\rho} \subseteq XT^{-1}_{1\text{-out}}(\text{d-}FSL_{\sigma,\rho})$.
(2) $\text{d-}XL_{\sigma,\rho} \subseteq \text{d-}XT^{-1}_{1\text{-out}}(\text{d-}FSL_{\sigma,\rho})$.

Proof. Let $\mathscr{A}$ be an X-automaton with input alphabet Σ and state set Q; let $\mathscr{D}$ be a family of state sets for $\mathscr{A}$.

Consider the infinitary language $K(\sigma, \rho)$ consisting of the ω-words over Q that are (σ, ρ)-accepting sequences with respect to $\mathscr{D}$. It belongs to $\text{d-}FSL_{\sigma,\rho}$ because it is (σ, ρ)-accepted by the deterministic finite-state automaton $\mathscr{B}=(Q \cup \{q^0\}, Q, \delta_{\mathscr{B}}, q^0, c0)$ with $q^0 \notin Q$ and $\delta_{\mathscr{B}}=\{(q', q, true, q, \Lambda) \mid q' \in Q \cup \{q^0\}, q \in Q\}$. In fact, one easily sees that

for $r \in Q^\omega$, the state sequence of the corresponding run r_1 of $\mathscr{B}$ on r satisfies $ran(r_1) = ran(r) \cup \{q^0\}$ and $inf(r_1) = inf(r)$. This makes it clear that $K(\sigma, \rho) = L_{\sigma,\rho}(\mathscr{B}, \mathscr{D}^0)$ with $\mathscr{D}^0 = \{D \cup \{q^0\} \mid D \in \mathscr{D}\}$ if (σ, ρ) equals $(ran, =)$ or $(ran, \subseteq)$, and $K(\sigma, \rho) = L_{\sigma,\rho}(\mathscr{B}, \mathscr{D})$ in the four remaining cases.

Modify $\mathscr{A}$ such that at each step it outputs its state, i.e., take output alphabet Q and replace every transition $(q, a, \beta, q', \varphi)$ by $(q, a, \beta, q', \varphi, q)$. This gives a (1-output) X-transducer $\mathscr{M}$ that maps each ω-word u over Σ onto the state sequences of the runs of $\mathscr{A}$ on u. u is (σ, ρ)-accepted by $\mathscr{A}$ if one of these state sequences belongs to $K(\sigma, \rho)$. Hence, $L_{\sigma,\rho}(\mathscr{A}, \mathscr{D}) = \{u \in \Sigma^\omega \mid$ there exists $(u, r) \in T(\mathscr{M})$ such that $r \in K(\sigma, \rho)\} = T(\mathscr{M})^{-1}(K(\sigma, \rho))$.

Note that $\mathscr{M}$ is deterministic when $\mathscr{A}$ is deterministic. □

Conversely, given an ω-preserving transducer and an automaton (using some sort of storage), we can compose their operations into a new automaton which uses the product of the storage types of the transducer and the automaton.

Lemma 3.2. *Let X_1 and X_2 be two storage types.*

(1) $X_1 T_\omega^{-1}(X_2 L_{\sigma,\rho}) \subseteq (X_1 \times X_2) L_{\sigma,\rho}$.

(2) d-$X_1 T_\omega^{-1}$(d-$X_2 L_{\sigma,\rho}$) $\subseteq$ d-$(X_1 \times X_2) L_{\sigma,\rho}$.

Proof. Let $\mathscr{M} = (Q_1, \Sigma_1, \delta_1, q_{in,1}, c_{in,1}, \Sigma_2)$ be an ω-preserving X_1-transducer, let $\mathscr{A} = (Q_2, \Sigma_2, \delta_2, q_{in,2}, c_{in,2})$ be an X_2-automaton, and let $\mathscr{D} \subseteq 2^{Q_2}$. We prove the lemma by constructing an $(X_1 \times X_2)$-automaton $\mathscr{B}$ that (σ, ρ)-accepts the ω-language $T(\mathscr{M})^{-1}(L_{\sigma,\rho}(\mathscr{A}, \mathscr{D})) = \{v \in \Sigma_1^\omega \mid (v, u) \in T(\mathscr{M})$ for some $u \in L_{\sigma,\rho}(\mathscr{A}, \mathscr{D})\}$. This is done using a straightforward direct product construction in which we simulate $\mathscr{M}$ and $\mathscr{A}$ in two alternating phases: first we simulate $\mathscr{M}$ until it produces some nonempty output and we store this output in the states of $\mathscr{B}$, then we simulate $\mathscr{A}$ on this output.

Formally, let m be a constant such that $\mathscr{M}$ never writes more than m symbols onto its output tape in a single transition, and let $\mathscr{B} = (Q, \Sigma_1, \delta, q_{in}, c_{in})$, where $Q = Q_1 \times Q_2 \times \{x \in \Sigma_2^* \mid |x| \leqslant m\}$, $q_{in} = (q_{in,1}, q_{in,2}, \Lambda)$, $c_{in} = (c_{in,1}, c_{in,2})$, and δ is composed as follows:

(simulation of $\mathscr{M}$) $((q_1, q_2, \Lambda), a, \beta_1, (q_1', q_2, w), \varphi_1) \in \delta$ for each transition $(q_1, a, \beta_1, q_1', \varphi_1, w) \in \delta_1$ and each state $q_2 \in Q_2$, and

(simulation of $\mathscr{A}$) $((q_1, q_2, aw), \Lambda, \beta_2, (q_1, q_2', w), \varphi_2) \in \delta$ for each $q_1 \in Q_1$, $a \in \Sigma_2 \cup \{\Lambda\}$ and $w \in \Sigma_2^*$ with $1 \leqslant |aw| \leqslant m$, and each transition $(q_2, a, \beta_2, q_2', \varphi_2) \in \delta_2$.

Clearly, using this simulation, a run of $\mathscr{M}$ on an ω-word v with output u can be combined with a run of $\mathscr{A}$ on u, yielding a run of $\mathscr{B}$ on v. Similarly, a run of the new automaton $\mathscr{B}$ on an ω-word v can be decomposed into a run of $\mathscr{M}$ on v and a run of $\mathscr{A}$ on the output u of $\mathscr{M}$. This output is guaranteed to be infinite since $\mathscr{M}$ is ω-preserving; this is essential in order to have $(v, u) \in T(\mathscr{M})$.

We can use Lemma 2.8 to define a family $\mathscr{D}'_\rho$ of state sets for $\mathscr{B}$: let ψ be the projection on the second component [i.e., $\psi(\{(q_1, q_2, x)\}) = \{q_2\}$], and let R be $T(\mathscr{M})^{-1}$. Note that, due to the simulation of the steps of $\mathscr{M}$, the second coordinate of the new run has multiple copies of the same state; however, these repetitions do not change the range or the infinity set of the (projection on the second coordinate of the) run.

Note that if both $\mathscr{M}$ and $\mathscr{A}$ are deterministic, then so is $\mathscr{B}$. □

If we combine these two lemmas we obtain our basic result: a characterization of the families $XL_{\sigma,\rho}$ and d-$XL_{\sigma,\rho}$ in terms of X-transductions and finite state ω-languages.

As observed before Lemma 3.1, it expresses that X-automata are equivalent to compositions of an X-transducer and a finite-state automaton. Another way of viewing this result is to say that the ω-languages accepted by X-automata are exactly those ω-languages that are reducible to a finite-state ω-language by an XT_ω-reduction. However, one should note that, in general, $XL_{\sigma,\rho}$ is not closed under XT_ω-reductions. (By Lemma 3.2, it is closed under XT_ω-reductions under the rather strong assumption that $X \times X$ can be simulated by X, using the notion of simulation introduced in the next section.)

Theorem 3.3. (1) $XL_{\sigma,\rho} = XT_\omega^{-1}(FSL_{\sigma,\rho})$.
(2) d-$XL_{\sigma,\rho}$ = d-XT_ω^{-1}(d-$FSL_{\sigma,\rho}$).

Proof. Clearly, these statements are a consequence of Lemmas 3.1 and 3.2 (with $X_1 = X$, and $X_2 = FS$), using the obvious fact that $(X \times FS)L_{\sigma,\rho} = XL_{\sigma,\rho}$. □

For the acceptance conditions with $\rho \in \{\subseteq, \sqcap\}$ the family $FSL_{\sigma,\rho}$ can be replaced by a single ω-language. Intuitively, this ω-language models the acceptance condition; from the above "reduction point of view" the ω-language is complete in $XL_{\sigma,\rho}$ with respect to XT_ω-reductions. A similar result was obtained in [40] for ω-languages accepted by Turing machines, and (implicitly) in [21] for regular ω-languages.

Theorem 3.4. (1) $XL_{ran,\subseteq} = XT_\omega^{-1}(\{1^\omega\})$,

$XL_{ran,\sqcap} = XT_\omega^{-1}(\{0^*1.\{0,1\}^\omega\})$,

$XL_{inf,\subseteq} = XT_\omega^{-1}(\{\{0,1\}^*.1^\omega\})$, *and*

$XL_{inf,\sqcap} = XT_\omega^{-1}(\{(0^*1)^\omega\})$.

(2) *The same equalities hold for the families of deterministic* ω*-languages* d-$XL_{\sigma,\rho}$ *and deterministic transductions* d-XT_ω^{-1}.

Proof. The inclusions from right to left are clear from Theorem 3.3 and the fact that the four given ω-languages, can be accepted by deterministic finite-state automata using the four respective acceptance conditions (cf. Example 2.6).

Let us consider the converse inclusions. By Lemma 2.7, we may assume that an ω-language L in $XL_{\sigma,\rho}$ with $\rho \in \{\subseteq, \sqcap\}$ is (σ, ρ)-accepted by an X-automaton $\mathscr{A}$ with single state set D: $L = L_{\sigma,\rho}(\mathscr{A}, \{D\})$. As in the proof of Lemma 3.1, we change $\mathscr{A}$ into an X-transducer $\mathscr{M}$. Here, however, $\mathscr{M}$ has the output alphabet $\{0, 1\}$, rather than the state set of $\mathscr{A}$; at each step $\mathscr{M}$ outputs the letter 1 if its state belongs to D, and the letter 0 otherwise.

It is clear that $\mathscr{M}$ outputs an ω-word from $0^*1.\{0, 1\}^\omega$ if and only if during its run it enters at least *once* a state from D or, equivalently, if the run is $(ran, \sqcap)$-accepting. Hence, $L_{ran,\sqcap}(\mathscr{A}, \{D\}) = T(\mathscr{M})^{-1}(0^*1.\{0, 1\}^\omega)$. In the same way, we see that $\mathscr{M}$ outputs an ω-word from $(0^*1)^\omega$ if and only if during its run it enters some state from D *infinitely often* or, equivalently, if the run is $(inf, \sqcap)$-accepting: $L_{inf,\sqcap}(\mathscr{A}, \{D\}) = T(\mathscr{M})^{-1}((0^*1)^\omega)$. The remaining two acceptance types can be handled analogously. □

Using Theorem 3.3, the known inclusions for the families $FSL_{\sigma,\rho}$ (Proposition 2.10) can be carried over directly to the families $XL_{\sigma,\rho}$ for an arbitrary storage type X, without being forced to generalize all the proofs for FS ω-languages. Thus, we obtain the following main result. (The corresponding result for deterministic automata will be presented in Section 5.)

Theorem 3.5. $XL_{ran,\subseteq} \subseteq XL_{ran,\sqcap} = XL_{ran,=} = XL_{inf,\subseteq} \subseteq XL_{inf,\sqcap} = XL_{inf,=}$.

Proof. By Proposition 2.10, $FSL_{ran,\subseteq} \subseteq FSL_{ran,\sqcap}$. Hence, $XT_\omega^{-1}(FSL_{ran,\subseteq}) \subseteq XT_\omega^{-1}(FSL_{ran,\sqcap})$. Thus, by Theorem 3.3(1), $XL_{ran,\subseteq} \subseteq XL_{ran,\sqcap}$. The same argument holds for the other inclusions and equalities. □

We cannot conclude that the inclusions are strict in general, like for finite-state ω-languages (Proposition 2.10) or pushdown automata [6]. In fact, for certain storage types all six families $XL_{\sigma,\rho}$ are equal (e.g., for Turing machines, see [8]). In Section 3.3 we will give a sufficient condition on X for all six families to be equal. To state that result in a neat way, we will use the notion of simulation of storage types.

3.2. Simulation of storage types

In order to compare the strength of two storage types, we introduce a notion of simulation. Rather than requiring that every instruction and every test of one storage type can be simulated by a "subroutine" using the other storage (an approach which was taken in [11]) we use deterministic transductions to formalize simulation. This turns out to be more convenient to work with. We will show in Corollary 3.9 that the definition is strong enough to ensure a fact that intuitively should follow from any

notion of simulation: if the storage type X can be simulated by the storage type Y, then $XL_{\sigma,\rho}$ is included in $YL_{\sigma,\rho}$.

Definition 3.6. Let X and Y be storage types. X *is simulated by* Y, denoted $X \leqslant Y$, if $\text{d-}XT_* \subseteq \text{d-}YT_*$.

Thus, we require that, in the finitary case, deterministic X-transducers can be simulated by deterministic Y-transducers. Taking transducers rather than automata forces the simulation to be straightforward. All the usual simulations satisfy our definition. Thus, e.g., since a pushdown stack can be used to simulate a counter, we have $CTR \leqslant PD$. Similarly, $PD^2 \leqslant T$ and $T \leqslant PD^2$, where T is the storage type of a Turing machine work-tape (and $PD^2 = PD \times PD$ is the storage type having two pushdowns, cf. the end of Section 2.1). Also, clearly, $FS \leqslant X$ for every storage type X.

First we show that simulation can be carried over from the finitary to the infinitary case.

Lemma 3.7. *If* $\text{d-}XT_* \subseteq \text{d-}YT_*$, *then* $\text{d-}XT \subseteq \text{d-}YT$ *and* $\text{d-}XT_\omega \subseteq \text{d-}YT_\omega$.

Proof. Let $\mathcal{M}$ be a deterministic X-transducer, with set of states Q. Observe that for a deterministic transducer the behaviour on infinite words is determined by its finitary transduction. If u is an infinite input word, then there is an infinite run of $\mathcal{M}$ on u with output v (finite or infinite) if and only if there is an infinite sequence $\langle (x_i, y_i)\rangle_{i\in\mathbb{N}}$ of elements from $T_*(\mathcal{M}, Q)$ with $u = lub\langle x_i\rangle_{i\in\mathbb{N}}$ and $v = lub\langle y_i\rangle_{i\in\mathbb{N}}$.

By assumption there is a deterministic Y-transducer $\mathcal{M}_1$ (with set of states Q_1) and a state set $D \subseteq Q_1$ such that $T_*(\mathcal{M}, Q) = T_*(\mathcal{M}_1, D)$. Since $T_*(\mathcal{M}, Q) \subseteq T_*(\mathcal{M}_1, Q_1)$, the above observation implies that $T(\mathcal{M}) \subseteq T(\mathcal{M}_1)$. Hence, for infinite inputs, we wish to restrict the domain of $\mathcal{M}_1$ to that of $\mathcal{M}$. Thus, $\mathcal{M}_1$ should be restricted in such a way that it has a run on an infinite input u only if it accepts all prefixes of u. We first change $\mathcal{M}_1$ such that, when reading a new letter, it knows whether it has accepted the word read so far. Introduce a new state $\bar{q}$ for every $q \in Q_1$. Intuitively, the bar means that the transducer has been in a state of D, and since then has read Λ only. According to this intuition, change the finite control of $\mathcal{M}_1$ as follows:

- if $(q, \Lambda, \beta, q', \varphi, w)$ is a transition of $\mathcal{M}_1$ with $q \in D$, then replace it by $(q, \Lambda, \beta, \bar{q}', \varphi, w)$,
- if $(q, \Lambda, \beta, q', \varphi, w)$ is a transition of $\mathcal{M}_1$, then add the transition $(\bar{q}, \Lambda, \beta, \bar{q}', \varphi, w)$,
- if $(q, a, \beta, q', \varphi, w)$ is a transition of $\mathcal{M}_1$ with $a \neq \Lambda$, then add the transition $(\bar{q}, a, \beta, q', \varphi, w)$.

Let $\mathcal{M}_2$ be the so obtained transducer, with set of states $Q_1 \cup \bar{Q}_1$, where $\bar{Q}_1 = \{\bar{q} \mid q \in Q_1\}$. Finally, we change $\mathcal{M}_2$ by dropping all transitions $(q, a, \beta, q', \varphi, w)$ with $a \neq \Lambda$ and $q \notin (D \cup \bar{Q}_1)$, thus obtaining the deterministic Y-transducer $\mathcal{M}_3$ that satisfies the above restriction. For any infinite input word u, $\mathcal{M}_3$ has a run on u with output v (finite or infinite) if and only if $\mathcal{M}$ has a run on u with output v. This shows that $T(\mathcal{M}_3) = T(\mathcal{M})$, and that if $\mathcal{M}$ is ω-preserving, then so is $\mathcal{M}_3$. □

It is well known that the languages accepted by deterministic and nondeterministic automata can be related using homomorphisms – for finitary (context-free) languages this was first shown in the Chomsky–Schützenberger theorem; for ω-languages, see the "projection lemmas" used to characterize nondeterministic behaviour of finite-state automata [34], Turing machines [40], and transition systems [30]. Such a result is also valid in our framework, for every storage type. (This should be compared with the mode of acceptance used by Staiger and Wagner, where a projection lemma for pushdown automata seems to be missing, cf. the first open problem at the end of Section 2 in [32]. Note, however, that our "projections" are not necessarily ω-preserving.)

Lemma 3.8. $XL_{\sigma,\rho} = HOM(\text{d-}XL_{\sigma,\rho})$.

Proof. Taking $X_1 = FS$ and $X_2 = \mathrm{X}$ in Lemma 3.2(1) we get $FST_\omega^{-1}(XL_{\sigma,\rho}) \subseteq (FS \times X)L_{\sigma,\rho}$. Since, obviously, $(FS \times X)L_{\sigma,\rho} = XL_{\sigma,\rho}$, this implies that $XL_{\sigma,\rho}$ is closed under inverses of ω-preserving FS-transductions and, in particular, under homomorphisms (see Lemma 2.11). Consequently, $HOM(\text{d-}XL_{\sigma,\rho}) \subseteq XL_{\sigma,\rho}$.

In order to prove the converse inclusion, let $\mathscr{A}$ be an X-automaton with finite control δ and input alphabet Σ. We change $\mathscr{A}$ into a deterministic (real-time) X-automaton $\mathscr{A}'$ by replacing each transition $t = (q, a, \beta, q', \varphi)$ in δ by $(q, t, \beta, q', \varphi)$. This means that each transition now reads its own name from the input; δ is the input alphabet of $\mathscr{A}'$. It is clear that (for an arbitrary family of state sets $\mathscr{Q}$) $L_{\sigma,\rho}(\mathscr{A}, \mathscr{Q}) = h(L_{\sigma,\rho}(\mathscr{A}', \mathscr{Q}))$, where $h: \delta \to \Sigma \cup \{\Lambda\}$ is the homomorphism that maps $t = (q, a, \beta, q', \varphi)$ onto a. □

The following result justifies the notion of simulation of storage types we have defined above.

Corollary 3.9. *If* $X \leqslant Y$, *then* $\text{d-}XL_{\sigma,\rho} \subseteq \text{d-}YL_{\sigma,\rho}$ *and* $XL_{\sigma,\rho} \subseteq YL_{\sigma,\rho}$.

Proof. In Lemma 3.7 it was shown that $X \leqslant Y$ implies $\text{d-}XT_\omega \subseteq \text{d-}YT_\omega$. By Theorem 3.3(2), $\text{d-}XL_{\sigma,\rho} = \text{d-}XT_\omega^{-1}(\text{d-}FSL_{\sigma,\rho})$. Thus, $\text{d-}XT_\omega \subseteq \text{d-}YT_\omega$ implies $\text{d-}XL_{\sigma,\rho} \subseteq \text{d-}YL_{\sigma,\rho}$. Now, by Lemma 3.8, also $XL_{\sigma,\rho} \subseteq YL_{\sigma,\rho}$ follows. □

3.3. Equality of the six families

In order to give a sufficient condition for the equality of all the families in Theorem 3.5, we use the following result. It is based on the inclusion $FSL_{inf,\sqcap} \subseteq PDL_{ran,\subseteq}$, which was proved in [6] using the pushdown essentially as a counter.

The storage type *blind counter*, denoted by BC, is equal to the storage type counter (see Example 2.2), except that it has no predicate symbols (cf. [14], where it is called a *partially* blind counter).

Lemma 3.10. $XL_{inf,\sqcap} \subseteq (X \times BC)L_{ran,\subseteq}$.

Proof. By Theorem 3.4, $XL_{inf,\sqcap} = XT_\omega^{-1}(\{(0^*1)^\omega\})$, while, according to Lemma 3.2, $XT_\omega^{-1}(BCL_{ran,\subseteq}) \subseteq (X \times BC)L_{ran,\subseteq}$. Hence, in order to prove the lemma, it suffices to show that $(0^*1)^\omega \in BCL_{ran,\subseteq}$.

We construct a BC-automaton $\mathscr{A}$ that uses its (blind) counter to ensure that during its runs it can read any finite number of consecutive 0's, but not infinitely many consecutive 0's.

$\mathscr{A}$ has two states q_0 and q_1 and, for $i \in \{0, 1\}$, the transitions $(q_i, 0, true, q_0, decr)$ and $(q_i, 1, true, q_1, \Lambda)$, and the transition $(q_i, \Lambda, true, q_1, incr)$. The initial state of $\mathscr{A}$ is q_1. Take $\mathscr{D} = \{\{q_0, q_1\}\}$.

Hence, for each step on the letter 0, $\mathscr{A}$ decreases its counter. Whenever $\mathscr{A}$ reads the letter 1 it enters state q_1. In this state, before reading the next input letter, $\mathscr{A}$ guesses the number of 0's on the tape before the next 1, and increases its counter value by (at least) this amount. Consequently, $L_{ran,\subseteq}(\mathscr{A}, \mathscr{D}) = (0^*1)^\omega$. □

We now give the sufficient condition: if the storage type X can simulate an additional blind counter, then all $XL_{\sigma,\rho}$ are the same.

Theorem 3.11. *If* $X \times BC \leqslant X$, *then* $XL_{ran,\subseteq} = XL_{inf,\sqcap}$.

Proof. By Lemma 3.10, $XL_{inf,\sqcap} \subseteq (X \times BC)L_{ran,\subseteq}$. According to Corollary 3.9, $X \times BC \leqslant X$ implies that $(X \times BC)L_{ran,\subseteq} \subseteq XL_{ran,\subseteq}$ and, consequently, $XL_{inf,\sqcap} \subseteq XL_{ran,\subseteq}$. Equality (of these and the other families $XL_{\sigma,\rho}$) now follows from Theorem 3.5. □

Using this result we clearly reobtain the equality of the families of ω-languages (σ, ρ)-accepted by Turing machines as given in [8]. As observed after Definition 2.5, this differs from the results of [40], where a proper hierarchy for Turing machines is obtained (due to a slightly different definition of acceptance).

It is obvious that the storage type BC^* can simulate an additional blind counter (where BC^* is the union of all BC^n, $n \in \mathbb{N}$, see the end of Section 2.1). BC^*-automata are *blind multicounter automata*, i.e., automata of which the storage consists of an arbitrary number of blind counters. It follows from Theorem 3.11 that for these automata the six acceptance criteria have the same power, i.e, the six families $BC^*L_{\sigma,\rho}$ are the same. This has some consequences for Petri nets. It is explained in [38] how, concerning their (infinite) sequential behaviour, Petri nets can be seen as blind multicounter automata (see also [18, 14]). In fact, the places of a Petri net can be divided (by analyzing its "reachability tree") into bounded places (i.e. places with a uniform bound on the number of tokens at the place at any time) and unbounded places. Clearly, each unbounded place may be viewed as a blind counter, the tokens at the bounded places together may be viewed as the state, and the transitions of the net as the finite control of the BC^*-automaton (where the labels of the transitions are

viewed as input symbols). This should explain that the families $BC^*L_{\sigma,\rho}$ equal the families of Petri net ω-languages, with (σ, ρ)-acceptance with respect to bounded markings, and with Λ-labeled transitions allowed. Although these families were not (explicitly) compared in the literature (the Petri nets in [38] are assumed to be Λ-free, i.e., real-time), the inclusion r-$BC^*L_{inf,\sqcap} \subset BC^*L_{ran,\subseteq}$ has been shown in [5, Theorem 3].

A particular case of Lemma 3.10 is of independent interest (and will be used in the sequel). For $X = FS$ we get $FSL_{inf,\sqcap} \subseteq BCL_{ran,\subseteq}$. From this and Corollary 3.9 it follows that if $BC \leqslant X$ [in fact, if $(0^*1)^\omega \in XL_{ran,\subseteq}$] then the family $XL_{ran,\subseteq}$ (and, hence, each family $XL_{\sigma,\rho}$) contains all regular ω-languages. Note that $BC \leqslant X$ is a rather weak assumption on a storage type X; it is satisfied by all the usual storage types (except FS of course).

Theorem 3.12. *If* $BC \leqslant X$, *then* $FSL_{inf,\sqcap} \subseteq XL_{\sigma,\rho}$.

4. Real-time automata

Similar to the basic characterization in Section 3 we now obtain a characterization of the families r-$XL_{\sigma,\rho}$ in terms of X-transductions and finite-state ω-languages (Theorem 4.4). As before, we then use it to transfer the inclusions known for the families $FSL_{\sigma,\rho}$ directly to the families r-$XL_{\sigma,\rho}$. This gives a diagram as we have found in Section 3 for the families $XL_{\sigma,\rho}$: a "hierarchy" of three levels, and the two inclusions between these levels may or may not be strict.

As in Section 3 we study the strictness of these two inclusions. We obtain upper bounds (in topological terms) on the ω-languages in the bottom two levels of the real-time hierarchy (Lemma 4.7). Using these upper bounds, we show that the two inclusions are strict for each storage type (Theorem 4.9).

The arguments used in this section, especially those in the proof of Lemma 4.6, are applicable to a class of automata more general than real-time automata, viz., automata that do not have an infinite computation on a finite input – we say that these automata have *finite delay*. Thus, both the hierarchy results and the topological upper bounds that can be obtained for real-time automata can be shown using the same techniques for automata with finite delay.

In Section 4.3 we investigate the expressive power of real-time automata and automata with finite delay. On the one hand, we show in Theorem 4.17 that real-time automata are equivalent to automata with finite delay whenever the storage type is powerful enough to simulate an additional queue ("in real time"). On the other hand, we show that for the two-counter storage the real-time restriction is strictly less powerful than the limitation to finite delay (Theorem 4.19). Finite-delay automata may have the same power as or strictly less power than unrestricted automata, depending on the storage type (Theorem 4.21).

4.1. The basic characterization for real-time automata

As mentioned in the above introduction, we investigate the real-time automata together with a slightly more general class of automata.

Definition 4.1. An X-transducer $\mathscr{A}$ has *finite delay* if there is no infinite run of $\mathscr{A}$ on a finite input word.

We use the prefix f- in the same way as we have used the prefixes d- and r-, i.e., to indicate families of (infinitary) languages (or transductions) defined by automata that have finite delay. Obviously, every real-time automaton has finite delay; so, we have $\text{r-}XL_{\sigma,\rho} \subseteq \text{f-}XL_{\sigma,\rho} \subseteq XL_{\sigma,\rho}$. By Lemma 2.9, we have equality for the trivial storage type: $\text{r-}FSL_{\sigma,\rho} = \text{f-}FSL_{\sigma,\rho} = FSL_{\sigma,\rho}$.

Automata with finite delay were already considered in the context of ω-languages. In [4] BC*-automata (i.e., Petri nets) having our finite-delay property are called *prompt* nets. That paper, however, focuses on nets that are 1-prompt, i.e., nets in which Λ-transitions are allowed, but not two consecutive Λ-transitions in a firing sequence (run). Note that the "finite delay" in the title of [4] refers to a fairness notion!

We have a decomposition result similar to the ones presented in Section 3.

Lemma 4.2. *Let* $p \in \{\text{r}, \text{f}\}$. *Then* $p\text{-}XL_{\sigma,\rho} \subseteq p\text{-}XT^{-1}_{1\text{-out}}(\text{d-}FSL_{\sigma,\rho})$.

Proof. The construction used in the proof of Lemma 3.1 transforms an X-automaton $\mathscr{A}$ into a 1-output X-transducer $\mathscr{M}$ by requiring that the automaton outputs its state at each step. Clearly, this does not change the input behaviour of the automaton; so, $\mathscr{M}$ is real-time (has finite delay) if and only if $\mathscr{A}$ is real-time (has finite delay). □

As in Section 3 the reverse inclusion can be stated in a slightly strengthened formulation. This time we have to be careful in the statement (and the proof) concerning the output behaviour of the transducer in the real-time case. Recall from Section 2.1 that a storage type is blind if it has no predicate symbols.

Lemma 4.3. *Let* X_1 *and* X_2 *be two storage types.*

(1) $\text{f-}X_1 T^{-1}_{\omega}(\text{f-}X_2 L_{\sigma,\rho}) \subseteq \text{f-}(X_1 \times X_2)L_{\sigma,\rho}$,

(2) $\text{r-}X_1 T^{-1}_{1\text{-out}}(\text{r-}X_2 L_{\sigma,\rho}) \subseteq \text{r-}(X_1 \times X_2)L_{\sigma,\rho}$.

(3) *If* X_2 *is a blind storage type, then* $\text{r-}X_1 T^{-1}_{\omega}(\text{r-}X_2 L_{\sigma,\rho}) \subseteq \text{r-}(X_1 \times X_2)L_{\sigma,\rho}$.

Proof. (1): In the case of automata with finite delay the proof of Lemma 3.2 is valid.

(2) and (3): For real-time automata we cannot split the simulation of the X_1-transducer $\mathscr{M}$ and the X_2-automaton $\mathscr{A}$ as we did in the proof of Lemma 3.2, because this introduces Λ-transitions. Hence, we simulate in a single step (of an $X_1 \times X_2$-automaton $\mathscr{B}$) one step of $\mathscr{M}$ and $|w|$ steps of $\mathscr{A}$ on the output w of $\mathscr{M}$. This implies, however, that the intermediate configurations of $\mathscr{A}$ are not available to apply tests to.

Consequently, this construction will work only if either $\mathscr{M}$ is 1-output (so, $\mathscr{B}$ has to simulate only a single step of $\mathscr{A}$) or the storage type of $\mathscr{A}$ is blind (which means that $\mathscr{A}$ cannot perform any tests on its storage configurations). Since the intermediate states of $\mathscr{A}$ are important for the acceptance of a run, we use (as in the proof of Lemma 2.9) the states of $\mathscr{B}$ to store these states.

Formally, let $\mathscr{M}=(Q_1, \Sigma_1, \delta_1, q_{\text{in},1}, c_{\text{in},1}, \Sigma_2)$ be a real-time ω-preserving X_1-transducer and let $\mathscr{A}=(Q_2, \Sigma_2, \delta_2, q_{\text{in},2}, c_{\text{in},2})$ be a real-time X_2-automaton.

The real-time $X_1 \times X_2$-automaton $\mathscr{B}=(Q, \Sigma, \delta, q_{\text{in}}, c_{\text{in}})$ is constructed as follows: $Q=Q_1 \times Q_2 \times 2^{Q_2}$, $q_{\text{in}}=(q_{\text{in},1}, q_{\text{in},2}, \{q_{\text{in},2}\})$, and, for each $U, U' \subseteq Q_2$, δ contains the transition $((q_1, q_2, U), a, \beta_1 \wedge \beta_{2,1} \wedge \cdots \wedge \beta_{2,k}, (q_1', q_2', U'), \varphi_1 \cdot \varphi_{2,1} \cdots \varphi_{2,k})$ whenever there is a transition $(q_1, a, \beta_1, q_1', \varphi_1, a_1 \cdots a_k) \in \delta_1$, $a_i \in \Sigma_2$ for $1 \leqslant i \leqslant k$, and a sequence $q_{2,0}, q_{2,1}, \ldots, q_{2,k}$ of states in Q_2 such that $U'=\{q_{2,1}, \ldots, q_{2,k}\}$, $q_2=q_{2,0}$, $(q_{2,i-1}, a_i, \beta_{2,i}, q_{2,i}, \varphi_{2,i}) \in \delta$ for $1 \leqslant i \leqslant k$, and $q_2'=q_{2,k}$.

We stress again that in $\mathscr{B}$ the tests $\beta_{2,1}, \beta_{2,2}, \ldots, \beta_{2,k}$ are applied to a single configuration c_2 rather than to the configurations $c_2, \mu_2(\varphi_{2,1})(c_2), \ldots, \mu_2(\varphi_{2,1} \cdots \varphi_{2,k-1})(c_2)$, respectively. Hence, the construction will not work unless (i) $k=1$, which is the case when $\mathscr{M}$ is 1-output, or (ii) $\beta_{2,1}=\cdots=\beta_{2,k}=\mathit{true}$, which is satisfied when X_2 is a blind storage type.

We accept runs of $\mathscr{B}$ by considering the third component of the states entered (infinitely often) during the run. Formally, this can be justified by applying Lemma 2.8, with $\psi(\{(q_1, q_2, U)\})=U$, and $R=T(\mathscr{M})^{-1}$. □

As for the corresponding results of Section 3, we now combine the above two lemmas to obtain a characterization of the families f-$XL_{\sigma,\rho}$ and r-$XL_{\sigma,\rho}$ in terms of X-transductions and finite-state ω-languages.

Theorem 4.4. (1) f-$XL_{\sigma,\rho}$ = f-$XT_\omega^{-1}(FSL_{\sigma,\rho})$,
(2) r-$XL_{\sigma,\rho}$ = r-$XT_\omega^{-1}(FSL_{\sigma,\rho})$,
(3) dr-$XL_{\sigma,\rho}$ = dr-XT_ω^{-1}(d-$FSL_{\sigma,\rho}$).

Proof. (1): According to Lemma 4.2, and Lemma 4.3 (with $X_1=X$ and $X_2=FS$), we have f-$XL_{\sigma,\rho} \subseteq$ f-$XT_{1\text{-out}}^{-1}$(d-$FSL_{\sigma,\rho}$) $\subseteq$ f-XT_ω^{-1}(f-$FSL_{\sigma,\rho}$) $\subseteq$ f-$(X \times FS)L_{\sigma,\rho}$ = f-$XL_{\sigma,\rho}$ and, consequently, f-$XL_{\sigma,\rho}$ = f-$XT_\omega^{-1}(FSL_{\sigma,\rho})$. Note that we used the fact that f-$FSL_{\sigma,\rho}=FSL_{\sigma,\rho}$.

Similarly, one proves (2). Note that FS is a blind storage type.

In order to prove (3), observe that the constructions used in the proofs of Lemma 4.2 (i.e., the proof of Lemma 3.1) and Lemma 4.3 preserve determinism. □

This allows us, as in Section 3, to transfer the inclusions known for the families $FSL_{\sigma,\rho}$ directly to the families f-$XL_{\sigma,\rho}$ and r-$XL_{\sigma,\rho}$, using (1) and (2) of Theorem 4.4.

Lemma 4.5. *Let* $p \in \{\text{r}, \text{f}\}$. *Then*

$$p\text{-}XL_{ran,\subseteq} \subseteq p\text{-}XL_{ran,\sqcap} = p\text{-}XL_{ran,=} = p\text{-}XL_{inf,\subseteq} \subseteq p\text{-}XL_{inf,\sqcap} = p\text{-}XL_{inf,=}.$$

4.2. Topological upper bounds

Again we investigate when the remaining two inclusions are equalities and when they are strict. Perhaps somewhat surprisingly they turn out to be always strict. We use the following two lemmas to demonstrate this. They lead to topological upper bounds on the ω-languages that can be $(ran, \subseteq)$-accepted and $(inf, \subseteq)$-accepted by X-automata having finite delay. These can then be used to exhibit ω-languages that "separate" the acceptance types $(ran, \subseteq)$, $(inf, \subseteq)$, and $(inf, \sqcap)$.

In the context of infinite finitely branching structures, König's Lemma is a basic and important tool. It was used in [38] to show that if every finite prefix of an ω-word is the label of a firing sequence (run) of a given Λ-free Petri net, then the ω-word itself is the label of an infinite firing sequence of the net. From this it follows that the ω-languages in r-$BC^*L_{ran,\subseteq}$ are all closed in the topology on Σ^ω (like the result Landweber [20] obtained for d-$FSL_{ran,\subseteq}$). In a straightforward way, this result can be extended not only to arbitrary storage, but also to automata with finite delay.

Lemma 4.6. *Let $\mathscr{A}$ be an X-automaton with finite delay, with state set Q. Then* $L_{ran,\subseteq}(\mathscr{A}, \{Q\}) = adh(L_*(\mathscr{A}, Q))$.

Proof. Let $\mathscr{A} = (Q, \Sigma, \delta, q_{in}, c_{in})$ and let $L = L_*(\mathscr{A}, Q)$. Then, obviously, $L = pref(L)$; so, $adh(L) = lim(pref(L)) = lim(L)$. Note that $L_{ran,\subseteq}(\mathscr{A}, \{Q\})$ is the set of all ω-words on which there exists a run of $\mathscr{A}$, without additional requirements for the state sequence of the run.

Assume that $u \in L_{ran,\subseteq}(\mathscr{A}, \{Q\})$. This implies that $u[n] \in L$ for each $n \in \mathbb{N}$ and, consequently, $u \in lim(L) = adh(L)$.

To prove the inclusion $adh(L) \subseteq L_{ran,\subseteq}(\mathscr{A}, \{Q\})$, let $u \in adh(L)$. For each $n \in \mathbb{N}$ there exists a finite run of $\mathscr{A}$ on the prefix $u[n]$ of u; it has at least length n. We use König's Lemma to show that there exists an infinite run of $\mathscr{A}$ on u.

Define the sets E_n, $n \in \mathbb{N}$, of instantaneous descriptions of $\mathscr{A}$ that are reachable from the initial instantaneous description in n steps on a prefix of u, as follows.

$$E_0 = \{(q_{in}, \Lambda, c_{in})\}, \text{ and for } n \geqslant 1,$$

$$E_n = \{(q, x, c) \mid x \in pref(u), \text{ and } (q', x', c') \vdash_{\mathscr{A}} (q, x, c) \text{ for some } (q', x', c') \in E_{n-1}\}.$$

Clearly, each of these sets is finite and nonempty. According to König's Lemma, there exists an infinite sequence $\langle (q_n, x_n, c_n) \rangle_{n \in \mathbb{N}}$ such that $(q_0, x_0, c_0) \in E_0$ and, for $n \geqslant 1$, $(q_{n-1}, x_{n-1}, c_{n-1}) \vdash_{\mathscr{A}} (q_n, x_n, c_n)$; this is an infinite run of $\mathscr{A}$ on $v = lub\langle x_n \rangle_{n \in \mathbb{N}}$. Note that v cannot be finite because $\mathscr{A}$ has finite delay. Moreover, every x_n is a prefix of u and, hence, $u = v$.

Since every infinite run of $\mathscr{A}$ is $(ran, \subseteq)$-accepting with respect to $\{Q\}$, this proves $adh(L) \subseteq L_{ran,\subseteq}(\mathscr{A}, \{Q\})$. □

Lemma 4.6 links (*ran*, $\subseteq$)-accepted ω-languages to closed sets in the topology on Σ^ω, and using an additional argument it shows that (*inf*, $\subseteq$)-accepted ω-languages are countable unions of closed sets.

Lemma 4.7. (1) f-$XL_{ran,\subseteq} \subseteq \mathscr{F}$.
(2) f-$XL_{inf,\subseteq} \subseteq \mathscr{F}_\sigma$.

Proof. Both $\mathscr{F}$ and $\mathscr{F}_\sigma$ are closed under (finite) union. Hence, it suffices to consider ω-languages (σ, ρ)-accepted with respect to a single state set. Let $\mathscr{A} = (Q, \Sigma, \delta, q_{\text{in}}, c_{\text{in}})$ be an X-automaton with finite delay and let $D \subseteq Q$.

(1): Obviously, $L_{ran,\subseteq}(\mathscr{A}, \{D\}) = L_{ran,\subseteq}(\mathscr{A}|_D, \{D\})$, where $\mathscr{A}|_D$ is the automaton $\mathscr{A}$ restricted to states from D. Hence, the result is a consequence of Lemma 4.6 (and the observation that $\mathscr{A}|_D$ has finite delay whenever $\mathscr{A}$ has finite delay). Recall that the adherences are the sets in $\mathscr{F}$ (i.e., the closed sets), see Proposition 1.2.

(2): Each (*inf*, $\subseteq$)-accepting state sequence (with respect to $\{D\}$) of $\mathscr{A}$ can be divided into two parts: an initial part in which all states from Q may occur, followed by an infinite part that enters only states from D. Thus, $L_{inf,\subseteq}(\mathscr{A}, \{D\}) = \bigcup \{L_{ran,\subseteq}(\mathscr{A}(q,x,c), \{D \cup Q_{q,x,c}\}) \mid (q_{\text{in}}, \Lambda, c_{\text{in}}) \vdash^* (q,x,c)\}$, where $\mathscr{A}(q,x,c)$ equals $\mathscr{A}$, except that it has a new initial state q'_{in} and a new path leading from q'_{in} to q, reading x from the input, and transforming c_{in} into c – this path copies the computation $(q_{\text{in}}, \Lambda, c_{\text{in}}) \vdash^* (q,x,c)$. $Q_{q,x,c}$ is the set of states that are added to $\mathscr{A}$ in order to form the new path.

The above union is countable; this follows from the fact that the number of instantaneous descriptions reachable from $(q_{\text{in}}, \Lambda, c_{\text{in}})$ in n steps is finite for each n.

Consequently, by (1), every ω-language in f-$XL_{inf,\subseteq}$ is a countable union of closed sets, hence an $\mathscr{F}_\sigma$-set. □

Using Proposition 1.3 and the topological upper bounds from the above result, we find the ω-languages we are looking for.

Corollary 4.8. (1) $0^*1.\{0,1\}^\omega \in$ r-$XL_{inf,\subseteq} -$ f-$XL_{ran,\subseteq}$.
(2) $(0^*1)^\omega \in$ r-$XL_{inf,\sqcap} -$ f-$XL_{inf,\subseteq}$.

Proof. (1): According to Proposition 1.3 and Lemma 4.7, $0^*1.\{0,1\}^\omega \notin \mathscr{F} \supseteq$ f-$XL_{ran,\subseteq}$; however, $0^*1.\{0,1\}^\omega \in FSL_{inf,\subseteq} \subseteq$ r-$XL_{inf,\subseteq}$ (see Example 2.6).

(2): Similarly, $(0^*1)^\omega \notin \mathscr{F}_\sigma \supseteq$ f-$XL_{inf,\subseteq}$; however, $(0^*1)^\omega \in FSL_{inf,\sqcap} \subseteq$ r-$XL_{inf,\sqcap}$. □

These two ω-languages can now be used to show that the inclusions from Lemma 4.5 are strict for all storage types. This gives the following main result.

Theorem 4.9. *Let* $p \in \{\text{r}, \text{f}\}$. *Then*

$$p\text{-}XL_{ran,\subseteq} \subset p\text{-}XL_{ran,\sqcap} = p\text{-}XL_{ran,=} = p\text{-}XL_{inf,\subseteq} \subset p\text{-}XL_{inf,\sqcap} = p\text{-}XL_{inf,=}.$$

Apart from FS, as far as we know the only specific storage type that was studied for its real-time behaviour is BC^*. Real-time blind multicounter automata (i.e., Petri nets) were studied in [38], both for acceptance with respect to states (i.e., bounded places) and for acceptance with respect to storage configurations (i.e., markings). We reobtain the results of Valk for acceptance with respect to states. The results obtained for the latter way of accepting infinitary languages are incomparable to those we have found in Theorem 4.9.

We now ask ourselves whether the same methods can be used to show strictness of the hierarchy of families $XL_{\sigma,\rho}$ for specific storage types X. As an example, as mentioned at the end of Section 3.1, such a strict hierarchy has been obtained for PD-automata with Λ-transitions. In [6, Section 3.3] this was shown using the following result. Let L be any finitary nonregular context-free language, e.g., $L=\{a^n b^n \mid n\in\mathbb{N}\}$, and let d be a letter not occurring in L, then $L.d^\omega \in PDL_{inf,\subseteq} - PDL_{ran,\subseteq}$ and $(L.d)^\omega \in PDL_{inf,\sqcap} - PDL_{inf,\subseteq}$.

Unfortunately, one should realize that topological arguments are of no use in obtaining these counterexamples for the hierarchy $PDL_{\sigma,\rho}$. Both $L.d^\omega$ and $(L.d)^\omega$ belong to $\mathscr{G}_\delta$ (they are the limits of $Ld.d^*$ and $(Ld)^*$), whereas even $PDL_{ran,\subseteq}$ contains ω-languages outside $\mathscr{F}_\sigma \cup \mathscr{G}_\delta$ because of the inclusion $FSL_{inf,\sqcap} \subseteq PDL_{ran,\subseteq}$ ([6], or Theorem 3.12). Such a regular ω-language is $0.\{0,1\}^*.1^\omega \cup 1.(0^*1)^\omega$ (see [20] for this example and for topological characterizations of the families d-$FSL_{\sigma,\rho}$).

It is quite striking that the ω-languages used to show the strictness of the inclusions in the hierarchy $PDL_{\sigma,\rho}$ are structurally very similar to ones that can be used to separate the families in the real-time and finite-delay hierarchies: in Corollary 4.8 one may replace $0^*1.\{0,1\}^\omega$ by $0^*.1^\omega$; thus, $L.d^\omega$ and $(L.d)^\omega$ remind us of $0^*.1^\omega$ and $(0^*1)^\omega$. Note that $0^*.1^\omega$ is of a higher topological complexity than $0^*1.\{0,1\}^\omega$: $0^*1.\{0,1\}^\omega \in \mathscr{G} - \mathscr{F}$, whereas $0^*.1^\omega \in (\mathscr{F}_\sigma \cap \mathscr{G}_\delta) - (\mathscr{F} \cup \mathscr{G})$, cf. Proposition 1.3.

4.3. The power of real-time automata

We compare the families f-$XL_{\sigma,\rho}$ and r-$XL_{\sigma,\rho}$ (and $XL_{\sigma,\rho}$). We start by investigating under what conditions (and how) an ω-language accepted by an automaton with finite delay (or even an unrestricted automaton) can be accepted by a real-time automaton.

For real-time automata we need a stronger notion of simulation.

Definition 4.10. Let X and Y be storage types. *X is real-time simulated by Y*, denoted $X \leqslant_r Y$, if both $X \leqslant Y$ and dr-$XT_* \subseteq$ dr-YT_*.

Unfortunately, we cannot show that dr-$XT_* \subseteq$ dr-YT_* implies d-$XT_* \subseteq$ d-YT_*; so, we explicitly require that $\leqslant_r$ implies $\leqslant$. A stronger alternative to Definition 4.10 would be: there should be a transformation of deterministic X-transducers into deterministic Y-transducers that turns every X-transducer into an equivalent

Y-transducer, and that preserves the real-time property; however, the present definition is easier to state and is all we need.

As in Section 3, we transfer simulation to infinitary transductions and ω-languages.

Lemma 4.11. *If* dr-$XT_* \subseteq$ dr-YT_*, *then* dr-$XT \subseteq$ dr-YT *and* dr-$XT_\omega \subseteq$ dr-YT_ω.

Proof. Let $\mathcal{M}$ be a deterministic real-time X-transducer, with set of states Q. By assumption, there is a deterministic real-time Y-transducer $\mathcal{M}_1$ (with set of states Q_1) and a state set $D \subseteq Q_1$ such that $T_*(\mathcal{M}, Q) = T_*(\mathcal{M}_1, D)$. As in the proof of Lemma 3.7, we observe that $T(\mathcal{M}) \subseteq T(\mathcal{M}_1)$. Again, we wish to restrict (for infinite inputs) the domain of $\mathcal{M}_1$ to that of $\mathcal{M}$, and we change $\mathcal{M}_1$ such that it accepts all prefixes of its input. In the absence of Λ-transitions this can be done by simply dropping all transitions $(q, a, \beta, q', \varphi, w)$ with $q \notin D$ or $q' \notin D$, thus obtaining the deterministic real-time Y-transducer $\mathcal{M}_2$ which satisfies the above restriction. For any infinite input word u, $\mathcal{M}_2$ has a run on u with output v (finite or infinite) if and only if $\mathcal{M}$ has a run on u with output v. This shows that $T(\mathcal{M}_2) = T(\mathcal{M})$, and that if $\mathcal{M}$ is ω-preserving, then so is $\mathcal{M}_2$. □

Lemma 4.12. r-$XL_{\sigma,\rho} = HOM_{1\text{-out}}$(dr-$XL_{\sigma,\rho}$).

Proof. The proof is similar to that of Lemma 3.8. Here one shows, using Lemma 4.3(2), that r-$XL_{\sigma,\rho}$ is closed under inverses of real-time 1-output FS-transductions and, in particular, under length-preserving homomorphisms: $HOM_{1\text{-out}} \subseteq$ r-$FST_{1\text{-out}} \subseteq$ $\subseteq$ r-$FST^{-1}_{1\text{-out}}$, see Lemma 2.11(2). □

Corollary 4.13. *If* $X \leqslant_r Y$, *then* dr-$XL_{\sigma,\rho} \subseteq$ dr-$YL_{\sigma,\rho}$ *and* r-$XL_{\sigma,\rho} \subseteq$ r-$YL_{\sigma,\rho}$.

Proof. Similar to the proof of Corollary 3.9, we have [using Theorem 4.4(3)] dr-$XL_{\sigma,\rho}$ = dr-XT^{-1}_ω(d-$FSL_{\sigma,\rho}$) and r-$XL_{\sigma,\rho} = HOM_{1\text{-out}}$(dr-$XL_{\sigma,\rho}$). Hence, dr-$XT_\omega \subseteq$ dr-YT_ω implies both dr-$XL_{\sigma,\rho} \subseteq$ dr-$YL_{\sigma,\rho}$ and r-$XL_{\sigma,\rho} \subseteq$ r-$YL_{\sigma,\rho}$. □

We will show that every ω-language (*inf*, $\sqcap$)-accepted by some automaton can be (*inf*, $\sqcap$)-accepted by a real-time automaton, provided we extend the storage with a queue. This is not true for the acceptance types (*ran*, $\subseteq$) and (*inf*, $\subseteq$). However, starting with an automaton with finite delay we can prove an even stronger property. For each acceptance condition (σ, ρ), every ω-language (σ, ρ)-accepted by an automaton with finite delay can be (σ, ρ)-accepted by a real-time automaton, when the storage is extended with a queue.

We use Q to denote a formalization of the *queue* as a storage type. Analogously to the storage type PD the configurations of Q are finite words; it has instructions for adding a letter to the rear of the queue and for removing a letter from the front, and it has tests to determine the first letter of the queue.

We need the following closure property of the family of ω-languages (*inf*, $\sqcap$)-accepted by real-time X-automata. Analogous properties hold for other acceptance types.

Lemma 4.14. r-$XL_{inf,\sqcap}$ *is closed under intersection with* ω*-languages from* $FSL_{inf,\sqcap}$.

Proof. Since the acceptance types (*inf*, $\sqcap$) and (*inf*, $=$) are equivalent for real-time automata (Theorem 4.9), we may demonstrate the lemma for r-$XL_{inf,=}$ and $FSL_{inf,=}$. Recall that $FSL_{inf,=} = \text{r-}FSL_{inf,=}$.

Generalizing the lemma, given a real-time X_i-automaton $\mathscr{A}_i$, and a family $\mathscr{D}_i$ of state sets for $\mathscr{A}_i$, $i=1,2$, by an obvious direct product construction one obtains a real-time $(X_1 \times X_2)$-automaton $\mathscr{A}$ simulating $\mathscr{A}_1$ and $\mathscr{A}_2$ in parallel. We wish a run of $\mathscr{A}$ to be accepting if its state sequence, when projected onto the ith component, is (*inf*, $=$)-accepting with respect to $\mathscr{D}_i$ for both $i=1$ and $i=2$. To realize this, we use as a family of state sets for $\mathscr{A}$ exactly those sets that, when projected onto the ith component, belong to $\mathscr{D}_i$, $i=1, 2$. □

Lemma 4.15. (1) $XL_{inf,\sqcap} \subseteq \text{r-}(Q \times X)L_{inf,\sqcap}$.
(2) $\text{f-}XL_{\sigma,\rho} \subseteq \text{r-}(Q \times X)L_{\sigma,\rho}$.

Proof. (1): Given an X-automaton $\mathscr{A}$ with input alphabet Σ, we transform $\mathscr{A}$ into a real-time X-automaton $\mathscr{A}^r$ over the alphabet $\Sigma \cup \{\phi\}$ by changing every Λ-transition into a transition that reads ϕ (where ϕ is a letter not in Σ).

Additionally, we construct a real-time 1-output Q-transducer $\mathscr{M}$ that "delays" input: in each move it stores its input letter at the end of the queue, and it outputs nondeterministically either the first letter of the queue (while removing it) or the symbol ϕ.

According to Lemma 4.3(2), there exists a (real-time) $(Q \times X)$-automaton (*inf*, $\sqcap$)-accepting $T(\mathscr{M})^{-1}(L_{inf,\sqcap}(\mathscr{A}^r, \mathscr{D}))$. However, in general, this language strictly contains the original ω-language $L_{inf,\sqcap}(\mathscr{A}, \mathscr{D})$. This is a consequence of the fact that $L_{inf,\sqcap}(\mathscr{A}^r, \mathscr{D})$ may contain ω-words of the form $x.\phi^\omega$, $x \in (\Sigma \cup \{\phi\})^*$; these ω-words correspond to infinite runs of $\mathscr{A}$ on finite words that happen to have accepting state sequences (with respect to $\mathscr{D}$). The acceptance of such an ω-word by $\mathscr{A}^r$ would imply that $\pi_\Sigma(x).\Sigma^\omega \subseteq T(\mathscr{M})^{-1}(L_{inf,\sqcap}(\mathscr{A}^r, \mathscr{D}))$, while not necessarily $\pi_\Sigma(x).\Sigma^\omega \subseteq L_{inf,\sqcap}(\mathscr{A}, \mathscr{D})$. Here we have used π_Σ to denote the projection onto the alphabet Σ; it removes the symbol ϕ.

Hence, before applying the transduction $T(\mathscr{M})^{-1}$, we first intersect the ω-language $L_{inf,\sqcap}(\mathscr{A}^r, \mathscr{D})$ with the regular ω-language $(\phi^*.\Sigma)^\omega = (\Sigma \cup \{\phi\})^\omega - (\Sigma \cup \{\phi\})^*.\phi^\omega$; by Lemma 4.14, this intersection again yields an ω-language from r-$XL_{inf,\sqcap}$. According to Lemma 4.3(2), $L_{inf,\sqcap}(\mathscr{A}, \mathscr{D}) = T(\mathscr{M})^{-1}(L_{inf,\sqcap}(\mathscr{A}^r, \mathscr{D}) \cap (\phi^*.\Sigma)^\omega)$ belongs to r-$(Q \times X)L_{inf,\sqcap}$.

(2): We now have $L_{\sigma,\rho}(\mathscr{A}, \mathscr{D}) = T(\mathscr{M})^{-1}(L_{\sigma,\rho}(\mathscr{A}^r, \mathscr{D}))$. In fact, $\mathscr{A}^r$ does not accept words from $(\Sigma \cup \{\phi\})^*.\phi^\omega$ because $\mathscr{A}$ is an automaton with finite delay. □

Remark 4.16. The first statement of the above lemma cannot be generalized to less powerful acceptance types like $(ran, \subseteq)$ and $(inf, \subseteq)$. Intuitively, this is due to the fact that these acceptance types cannot force an automaton to perform a certain action (in the above proof: removing a letter from the queue) infinitely often.

Recall that, if $BC \leqslant X$, then all ω-languages accepted by FS-automata are included in $XL_{\sigma,\rho}$ for each acceptance type (Theorem 3.12). We then may use the counter-examples of Corollary 4.8: $0^*1.\{0, 1\}^\omega \in XL_{ran,\subseteq}$, whereas, for every storage type Y, $0^*1.\{0, 1\}^\omega \notin \text{f-}YL_{ran,\subseteq}$. Similarly, $(0^*1)^\omega \in XL_{inf,\subseteq}$, while $(0^*1)^\omega \notin \text{f-}YL_{inf,\subseteq}$.

In Section 3 we have shown that the hierarchy of the families $XL_{\sigma,\rho}$ collapses into a single family when the storage type X can simulate an extra blind counter (Theorem 3.11). Now Lemma 4.15 tells us that for each acceptance type automata with finite delay are equivalent to real-time automata, for storage types (like Q^*) that can simulate an additional queue in real time. More precisely, in this case the class of families r-$XL_{\sigma,\rho}$, f-$XL_{\sigma,\rho}$, and $XL_{\sigma,\rho}$ consists of precisely three families: r-$XL_{ran,\subseteq} \subset$ r-$XL_{inf,\subseteq} \subset$ r-$XL_{inf,\sqcap}$, as shown in Theorem 4.17.

Theorem 4.17. *If $Q \times X \leqslant_r X$ then the diagram of Fig. 2 holds.*

Proof. The strict inclusions from left to right for real-time (and finite delay) automata are presented in Theorem 4.9. We consider the equalities from the diagram.

Using Lemma 4.15, the assumption of the theorem, and Corollary 4.13, we have $XL_{inf,\sqcap} \subseteq$ r-$(Q \times X)L_{inf,\sqcap} \subseteq$ r-$XL_{inf,\sqcap}$ and f-$XL_{\sigma,\subseteq} \subseteq$ r-$(Q \times X)L_{\sigma,\subseteq} \subseteq$ r-$XL_{\sigma,\subseteq}$.

In order to show the equality of the families $XL_{\sigma,\rho}$, it suffices to demonstrate that $BC \times X \leqslant X$ (Theorem 3.11). First we observe that by definition $Q \times X \leqslant_r X$ implies $Q \times X \leqslant X$. Since, clearly, $BC \leqslant Q$ we have $BC \times X \leqslant Q \times X \leqslant X$. □

We have now seen some examples of storage types for which automata with finite delay are equivalent to real-time automata. In particular, we have the equalities f-$FSL_{\sigma,\rho}$ = r-$FSL_{\sigma,\rho}$ for the simple storage type FS as well as f-$XL_{\sigma,\rho}$ = r-$XL_{\sigma,\rho}$ for the more powerful storage types that satisfy $Q \times X \leqslant_r X$.

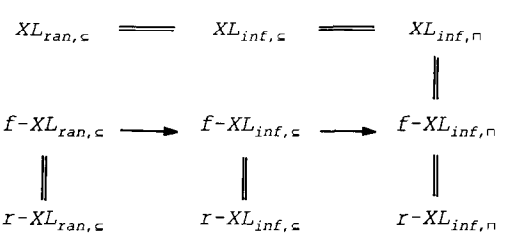

Fig. 2. Inclusion diagram in the case that $Q \times X \leqslant_r X$.

However, in general, the families f-$XL_{\sigma,\rho}$ and r-$XL_{\sigma,\rho}$ are not equal. We will show that the ω-language

$$BIN_\omega = \{x.a^k \mid x \in \{0, 1\}^*,\ k = nr(x)\}.b^\omega \cup \{0, 1\}^\omega,$$

where $nr(x)$ denotes the integer represented by $x \in \{0, 1\}^*$ as a binary number, can be accepted by an automaton with finite delay having two counters as storage but not by such an automaton that is real-time. This example is essentially the one given by Jantzen in [18] for finitary languages accepted by (real-time) Petri nets.

Lemma 4.18. $BIN_\omega \in$ f-$CTR^2 L_{ran,\subseteq} -$ r-$CTR^2 L_{inf,\sqcap}$.

Proof. (a) $BIN_\omega \in$ f-$CTR^2 L_{ran,\subseteq}$. We use $incr_i$ and $decr_i$ to denote the increment and decrement instructions for the ith counter ($i \in \{1, 2\}$).

Let $\mathscr{A}$ be the (deterministic) CTR^2-automaton with initial state p_1 and the following transitions:

$(p_1, 0, true, p_2, \Lambda)$, $(p_1, 1, true, p_2, incr_2)$,

$(p_2, \Lambda, \neg zero_1, p_2, decr_1\, incr_2\, incr_2)$, $(p_2, \Lambda, zero_1, p_3, \Lambda)$,

$(p_3, \Lambda, \neg zero_2, p_3, decr_2\, incr_1)$, $(p_3, \Lambda, zero_2, p_1, \Lambda)$,

$(p_i, a, \neg zero_1, p_4, decr_1)$, $i \in \{1, 4\}$, and $(p_i, b, zero_1, p_5, \Lambda)$, $i \in \{1, 4, 5\}$.

Let $Q = \{p_1, p_2, p_3, p_4, p_5\}$. One easily verifies that $L_{ran,\subseteq}(\mathscr{A}, \{Q\}) = BIN_\omega$: the first counter represents the value $nr(y)$, where y is the prefix (in $\{0, 1\}^*$) of the input read, the second counter is used (in states p_2 and p_3) to multiply the first counter by two. We still have to show that $\mathscr{A}$ has finite delay. Observe that any sequence of Λ-moves of $\mathscr{A}$ must – from some moment on – either take place in state p_2 or in state p_3. However, the number of successive applications of the instructions $decr_1\, incr_2\, incr_2$ ($decr_2\, incr_1$) is bounded by the value of the first (second) counter. Hence, infinite sequences of Λ-moves are impossible.

(b) $BIN_\omega \notin$ r-$CTR^2 L_{inf,\sqcap}$. The argument closely follows [18]. Assume that $BIN_\omega = L_{inf,\sqcap}(\mathscr{A}, \{D\})$, where $\mathscr{A}$ is a real-time CTR^2-automaton with state set Q. Let m be the maximal value that can be added to the counters in a single step of $\mathscr{A}$. Thus, if $\mathscr{A}$ reads a word $x \in \{0, 1\}^*$ from its input tape with $|x| = n$, then the total value of the counters is at most $n \cdot m$. Consequently, there exists a constant c such that there are at most $c \cdot n^2$ possibilities for the storage configuration of $\mathscr{A}$ after reading a word of length n.

Choose $n_0 \in \mathbb{N}$ such that $2^{n_0} > \#Q \cdot c \cdot n_0^2$.

For each word $x \in \{0, 1\}^*$ with $|x| = n_0$ there exists an infinite run $r(x) = \langle (q_i(x), v[i], c_i(x)) \rangle_{i \in \mathbb{N}}$ of $\mathscr{A}$ on $v = xa^{nr(x)}b^\omega$ that is (inf, $\sqcap$)-accepting with respect to $\{D\}$. Since there are 2^{n_0} words of length n_0 in $\{0, 1\}^*$ but less possibilities for the pairs $(q_{n_0}(x), c_{n_0}(x))$, there are two different words x_1, x_2 of length n_0 over $\{0, 1\}$ – say with $nr(x_1) < nr(x_2)$ – such that $(q_{n_0}(x_1), c_{n_0}(x_1)) = (q_{n_0}(x_2), c_{n_0}(x_2))$. Following the

definition of $r(x_2)$, starting in $(q_{n_0}(x_2), x_2, c_{n_0}(x_2))$ $\mathscr{A}$ may read $a^{nr(x_2)}b^\omega$ from its input while entering some state in D infinitely often. By a combination of the two runs $r(x_1)$ and $r(x_2)$ [switching from the one to the other when reaching state $q_{n_0}(x_1)=q_{n_0}(x_2)$ and configuration $c_{n_0}(x_1)=c_{n_0}(x_2)$], we obtain a run of $\mathscr{A}$ on $x_1a^{nr(x_2)}b^\omega$ which is (*inf*, $\sqcap$)-accepting with respect to $\{D\}$. This contradicts our assumption since $x_1a^{nr(x_2)}b^\omega \notin BIN_\omega$. This proves $BIN_\omega \notin$ r-$CTR^2L_{inf,\sqcap}$. □

Hence, BIN_ω is an ω-language that distinguishes the real-time two-counter automata from the two-counter automata with finite delay.

Theorem 4.19. r-$CTR^2L_{\sigma,\rho} \subset$ f-$CTR^2L_{\sigma,\rho}$.

Remark 4.20. Note that the same line of reasoning can be used to show that $BIN_\omega \notin$ r-$CTR^*L_{inf,\sqcap}$.

Lemma 4.18 can also be proved for the storage type BC^2. One then uses the ω-language $BIN'_\omega = \{w.a^k \mid w \in \{0, 1\}^*,\ 0 \leqslant k \leqslant nr(w)\}.b^\omega \cup \{0,1\}^\omega$ (see again [18]). In fact, $BIN'_\omega \in$ f-$BC^2L_{ran,\subseteq}$ − r-$BC^*L_{inf,\sqcap}$.

From Theorem 3.11, Corollary 4.8, and the above remark on BIN'_ω, it now follows that the diagram of Fig. 3 holds for Petri nets (with acceptance with respect to bounded places, see [38] for real-time nets). We conjecture that the inclusion indicated by $\Rightarrow$ is an equality.

A diagram with the same inclusions and equalities holds for the storage types CTR^* and CTR^2.

Each of the inclusions r-$XL_{\sigma,\rho} \subseteq$ f-$XL_{\sigma,\rho} \subseteq XL_{\sigma,\rho}$ may be either an equality (as, e.g., for $X=FS$) or may be strict (as for $X=BC^*$), except perhaps the inclusion f-$XL_{inf,\sqcap} \subseteq XL_{inf,\sqcap}$ for which no example of strictness has yet been given. We do this now.

Theorem 4.21. *There is a storage type Z such that* f-$ZL_{inf,\sqcap} \subset ZL_{inf,\sqcap}$.

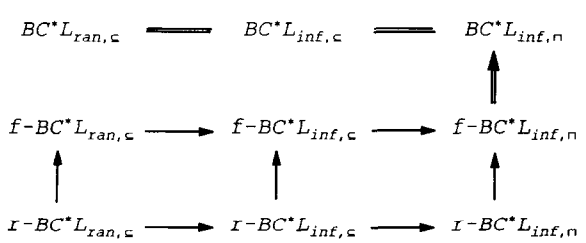

Fig. 3. Inclusion diagram for the storage type BC^*.

Proof. Let Z be the rather exotic storage type $(\mathbb{N} \times \mathbb{N} \times \{\uparrow, \downarrow\}, \{(1, 1, \uparrow)\}, \{zero\}, \{incr, decr_1, decr_2, half_1\}, \mu)$, where

$$\mu(zero)(r, s, d) = true \text{ if and only if } r = s = 0,$$

$$\mu(incr)(r, s, d) = (r+1, s+1, d) \quad \text{if } d = \uparrow, \text{ and undefined otherwise,}$$

$$\mu(decr_1)(r, s, d) = (r-1, s, \downarrow) \quad \text{if } r \geqslant 1, \text{ and undefined otherwise,}$$

$$\mu(decr_2)(r, s, d) = (r, s-1, \downarrow) \quad \text{if } s \geqslant 1, \text{ and undefined otherwise,}$$

$$\mu(half_1)(r, s, d) = (\tfrac{1}{2}r, s, \downarrow) \quad \text{if } r \text{ is even, and undefined otherwise.}$$

Note that Z is a "one-turn" two-counter storage, in the sense that the counters can be incremented (synchronously) in the first phase, and can only be decremented in the second phase of a run.

Consider the following variant of BIN_ω:

$$BIN_\omega^{rev} = \{x.a^k \mid x \in \{0, 1\}^*, k = nr(x^{rev})\}.b^\omega,$$

where x^{rev} is the mirror image of x.

A Z-automaton $\mathscr{A}$ $(inf, \sqcap)$-accepting BIN_ω^{rev} can be constructed in the following way. $\mathscr{A}$ starts by nondeterministically guessing a value for k, and puts this value in both its counters. Then it checks whether this value is represented by x^{rev} in binary: on input 0 it halves the first counter, on input 1 it does the same after first decrementing the counter by one. Finally, using its second counter, it checks whether the number of a's on the input tape also equals k. After reading the last a, the zero test is performed. Obviously, $\mathscr{A}$ does not have finite delay because it has an infinite run on Λ (guessing an infinite value for k).

BIN_ω^{rev} cannot be $(inf, \sqcap)$-accepted by a Z-automaton with finite delay. For each Z-automaton with finite delay there is a fixed bound on the number that can be added to the counters in a computation that uses only Λ-transitions. In fact, if such a bound would not exist, there would be arbitrary long sequences of Λ-transitions incrementing the counter. Consequently, there would be a cycle of such transitions in the automaton, contradicting the finite-delay property. Now that we have obtained the bound, the proof is similar to the proof of Lemma 4.18: the contents of the counters after reading x can be bounded by a linear function in $|x|$. We leave the details to the reader. □

5. Deterministic automata

By now the reader will have guessed the type of results we want to derive in this section. Using the characterization result for deterministic automata given in Theorem 3.3, we obtain a diagram for the families d-$XL_{\sigma,\rho}$, which is similar to the well-known diagram for the families d-$FSL_{\sigma,\rho}$ (which was presented in Proposition 2.10). As in the previous sections, this does not yet give any information concerning

the strictness of the inclusions. Like for real-time automata, we give simple topological upper bounds on the families d-$XL_{\sigma,\rho}$ which are then used to obtain strictness of the inclusions (Theorem 5.5). After comparing the strength of deterministic and nondeterministic automata in Theorem 5.6, we close the section by studying a property of storage types related to the closure under complement of the families d-$XL_{\sigma,=}$.

For deterministic automata there are some well-known relations between acceptance type and language-theoretic operations.

Lemma 5.1. *Let $\mathcal{A}$ be a deterministic X-automaton with state set Q and alphabet Σ, and let $\mathcal{D} \subseteq 2^Q$. Then*

(1) $L_{ran,\subseteq}(\mathcal{A}, \{Q\}) = adh(L_*(\mathcal{A}, Q))$,

(2) $L_{ran,\sqcap}(\mathcal{A}, \mathcal{D}) = L_*(\mathcal{A}, \bigcup \mathcal{D}).\Sigma^\omega \cap adh(L_*(\mathcal{A}, Q))$, and

(3) $L_{inf,\sqcap}(\mathcal{A}, \mathcal{D}) = lim(l_*(\mathcal{A}, \bigcup \mathcal{D}))$.

Proof. (1): $L_{ran,\subseteq}(\mathcal{A}, \{Q\}))$ is the set of all ω-words over Σ on which there exists a run of $\mathcal{A}$. Clearly, each of these ω-words belongs to $adh(L_*(\mathcal{A}, Q))$. The reverse inclusion is also obvious. If $\mathcal{A}$ may read arbitrary long prefixes of an ω-word u, then there exists an infinite run of $\mathcal{A}$ on u because $\mathcal{A}$ is deterministic. (See also the proofs of Lemmas 3.7 and 4.6.)

(2): $L_{ran,\sqcap}(\mathcal{A}, \mathcal{D})$ consists of all ω-words on which there exists a run of $\mathcal{A}$, and, additionally, for which this (unique) run enters, at least once, a state from one of the sets from $\mathcal{D}$. These ω-words form the set $adh(L_*(\mathcal{A}, Q)) \cap L_*(\mathcal{A}, \bigcup \mathcal{D}).\Sigma^\omega$.

(3): The proof is similar to that of (1) and (2). We only require that the run enters infinitely often a state from $\bigcup \mathcal{D}$. □

The above result enables us to give topological upper bounds on the ω-languages that are accepted by deterministic automata and, consequently, to give examples of elementary ω-languages that cannot be accepted by any deterministic automaton using a specific acceptance type.

Given two families $\mathcal{K}$ and $\mathcal{L}$ of ω-languages we use $\mathcal{K} \wedge \mathcal{L}$ to denote $\{K \cap L \mid K \in \mathcal{K}, L \in \mathcal{L}\}$, and $\mathcal{B}(\mathcal{K})$ to denote the Boolean closure of the family $\mathcal{K}$. Note that $\mathcal{B}(\mathcal{F}) = \mathcal{B}(\mathcal{G})$ and $\mathcal{B}(\mathcal{F}_\sigma) = \mathcal{B}(\mathcal{G}_\delta)$.

Lemma 5.2. (1) d-$XL_{ran,\subseteq} \subseteq \mathcal{F}$.

(2) d-$XL_{ran,\sqcap} \subseteq \mathcal{F} \wedge \mathcal{G}$.

(3) d-$XL_{ran,=} \subseteq \mathcal{B}(\mathcal{F})$.

(4) d-$XL_{inf,\subseteq} \subseteq \mathcal{F}_\sigma$.

(5) d-$XL_{inf,\sqcap} \subseteq \mathcal{G}_\delta$.

(6) d-$XL_{inf,=} \subseteq \mathcal{B}(\mathcal{F}_\sigma)$.

Proof. (1), (2), and (5) are clear from Lemma 5.1 and the relation between the language-theoretic operations and topological families (Proposition 1.2).

(4) can be shown just as the inclusion f-$XL_{inf,\subseteq} \subseteq \mathscr{F}_\sigma$ in Lemma 4.7(2), using (1).

To show (3) and (6), consider an arbitrary deterministic X-automaton $\mathscr{A}$ and a family $\mathscr{D}$ of state sets for $\mathscr{A}$. Then $L_{\sigma,=}(\mathscr{A}, \mathscr{D}) = \bigcup_{D\in\mathscr{D}} [L_{\sigma,\subseteq}(\mathscr{A}, \{Q\}) - L_{\sigma,\sqcap}(\mathscr{A}, \{Q-D\})]$. Now d-$XL_{ran,=} \subseteq \mathscr{B}(\mathscr{F})$ and d-$XL_{inf,=} \subseteq \mathscr{B}(\mathscr{F}_\sigma)$ follow from (1), (2), (4) and (5). □

Corollary 5.3. (1) $0^*1.\{0,1\}^\omega \in$ dr-$XL_{ran,\sqcap} -$ d-$XL_{ran,\subseteq}$.

(2) $(0^*1)^\omega \in$ dr-$XL_{inf,\sqcap} -$ d-$XL_{inf,\subseteq}$.

(3) $\{0,1\}^*.1^\omega \in$ dr-$XL_{inf,\subseteq} -$ d-$XL_{inf,\sqcap}$.

Proof. Clear from Example 2.6, Lemma 5.2, and Proposition 1.3. □

Lemma 5.4. $\{0,1\}^\omega - 0^*1.0^\omega \in$ dr-$XL_{ran,=} -$ d-$XL_{ran,\sqcap}$.

Proof. Let $K = \{0,1\}^\omega - 0^*1.0^\omega = 0^\omega \cup 0^*10^*1.\{0,1\}^\omega$.

Let $\mathscr{A}$ be the deterministic real-time FS-automaton with state set $Q = \{q_0, q_1, q_2\}$ and transitions $(q_i, 0, \mathit{true}, q_i, \Lambda)$ for $i\in\{0,1,2\}$, $(q_i, 1, \mathit{true}, q_{i+1}, \Lambda)$ for $i\in\{0,1\}$, and $(q_2, 1, \mathit{true}, q_2, \Lambda)$. If $\mathscr{D} = \{\{q_0\}, Q\}$, then $L_{ran,=}(\mathscr{A}, \mathscr{D}) = K$.

On the other hand, assume that $K\in$ d-$XL_{ran,\sqcap}$. According to Lemma 5.2(2), d-$XL_{ran,\sqcap} \subseteq \mathscr{F} \wedge \mathscr{G}$. Thus, assume that K is of the form $adh(L_1)\cap L_2.\{0,1\}^\omega$ for finitary languages L_1 and L_2. Since $0^\omega \in K \subseteq L_2.\{0,1\}^\omega$, we have $0^n \in L_2$ for some $n\in\mathbb{N}$. On the other hand, $0^n10^*1.0^\omega \subseteq K \subseteq adh(L_1)$, so $\mathit{pref}(0^n10^*) \subseteq \mathit{pref}(L_1)$. Consequently, $0^n1.0^\omega \subseteq adh(L_1)\cap L_2.\{0,1\}^\omega = K$; a contradiction. □

We now present the next main result, the full diagram for the families d-$XL_{\sigma,\rho}$.

Theorem 5.5. (1) *The diagram of Fig. 4 holds.*

(2) *The same diagram holds for the families* dr-$XL_{\sigma,\rho}$.

Proof. (1): The inclusions ($\subseteq$ rather than $\subset$) for the families d-$XL_{\sigma,\rho}$ can be obtained from the finite-state case (Proposition 2.10) using our characterization in terms of inverse transductions (Theorem 3.3(2)). The strictness of the inclusions d-$XL_{ran,\subseteq} \subseteq$ d-$XL_{ran,\sqcap} \subseteq$ d-$XL_{ran,=}$, and the incomparability of d-$XL_{inf,\subseteq}$ and d-$XL_{inf,\sqcap}$ follow from Corollary 5.3 and Lemma 5.4.

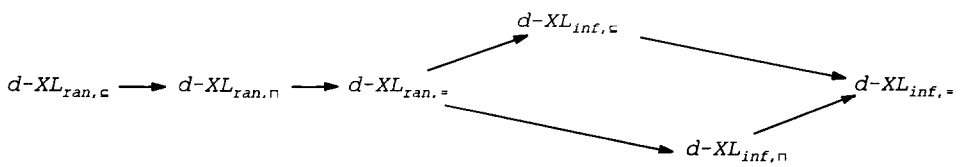

Fig. 4. Inclusion diagram for deterministic automata.

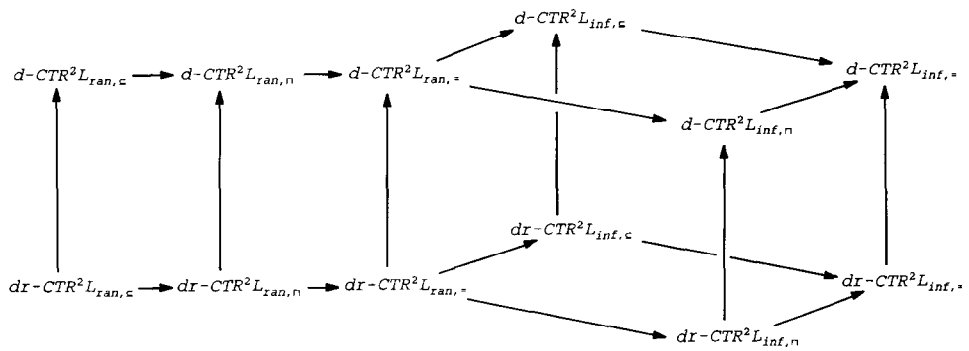

Fig. 5. Inclusion diagram for CTR^2, also valid for CTR^*.

(2): For dr-$XL_{\sigma,\rho}$ the inclusions can be obtained using the characterization in Theorem 4.4(3). □

The diagram for the families d-$XL_{\sigma,\rho}$ and the one for the families dr-$XL_{\sigma,\rho}$ can be combined into one figure. In general, the inclusion dr-$XL_{\sigma,\rho} \subseteq$ d-$XL_{\sigma,\rho}$ can be an equality as for FS [Lemma 2.9(2)] and for the storage type U which is studied in Section 6 (see Theorem 6.3). Since the ω-language BIN_ω given in Section 4 belongs to d-$CTR^2L_{ran,\subseteq}$ but not to r-$CTR^2L_{inf,=}$ (see Lemma 4.18 and its proof), the diagram of Fig. 5 holds for the storage type CTR^2 (as well as for CTR^*).

The infinitary behaviour of deterministic machines was studied in several places. We mention pushdown automata [23, 7], Turing machines [40, 8], and Petri nets [5].

Unfortunately, the results that were obtained in these papers concerning the relative strength of the acceptance types do not follow directly from Theorem 5.5, because several different choices with respect to determinism and acceptance were made. We mention some of the differences.

In [7] pushdown automata are required to be *total* ("have the *continuity property*"), in the "global" sense that they should have an infinite run on each possible input (reading all letters). It is explained in that paper that this influences only the class d-$PDL_{ran,\sqcap}$. In fact, with this requirement d-$PDL_{ran,\subseteq}$ and d-$PDL_{ran,\sqcap}$ become "complementary" families (see the discussion preceding Definition 5.7). Thus, with the exception of the family d-$PDL_{ran,\sqcap}$, our families (and the relations between them) are the same as those of Cohen and Gold.

As explained before, the acceptance conditions in [40] do not require a machine to read all its input. Nevertheless, the relations (but not necessarily the families) obtained by Wagner and Staiger for deterministic Turing machines are the same as the ones from Theorem 5.5, except again for the acceptance type (*ran*, ⊓).

For Petri nets our definition of determinism is not the usual one. In general, the definition of determinism can be weakened in a natural way. A first alternative is to require that for no configuration c there are two transitions $(q, a_i, \beta_i, q_i', \varphi_i, w_i)$, $i=1, 2$, with $a_1 = a_2$ or $a_1 = \Lambda$, such that, for both $i=1$ and $i=2$, $\mu(\beta_i)(c) = true$ *and* $\mu(\varphi_i)(c)$ *is*

defined. For Petri nets (cf. [5]) this alternative is further weakened by restricting the requirement to configurations c that can be reached from the initial configuration of the automaton. Although this notion of determinism is weaker than the one we use, it is still stronger than a "global" notion of determinism which requires that there is at most one infinite run on each input.

Rather than confusing the reader with several notions of determinism treated at the same time, we have chosen to illustrate our techniques using one, classical, definition. We believe that similar results can be obtained for other, "reasonable", definitions of determinism, since the constructions used in the proof of our main characterization result (i.e., in the proofs of Lemmas 3.1 and 3.2) do not introduce nondeterminism.

For finite-state automata the expressive power of determinism can be read from Proposition 2.10: deterministic automata are as powerful as nondeterministic automata for the acceptance types $(ran, \subseteq)$, $(inf, \subseteq)$, and $(inf, =)$, but they are less powerful for the remaining three acceptance types. We compare deterministic and nondeterministic automata in the next result.

Equivalence for the acceptance types $(ran, \subseteq)$ and $(inf, \subseteq)$ turns out to be a rare property of storage types.

Theorem 5.6. (1) d-$XL_{ran,\sqcap} \subset XL_{ran,\sqcap}$, d-$XL_{ran,=} \subset XL_{ran,=}$, *and* d-$XL_{inf,\sqcap} \subset XL_{inf,\sqcap}$.
(2) *If* $BC \leqslant X$, *then, additionally,* d-$XL_{ran,\subseteq} \subset XL_{ran,\subseteq}$, *and* d-$XL_{inf,\subseteq} \subset XL_{inf,\subseteq}$.

Proof. (1): Follows quite easily from Theorems 5.5 and 3.5.

(2): The strictness of these inclusions follows from the fact that d-$XL_{ran,\subseteq}$ and d-$XL_{inf,\subseteq}$ do not contain all regular ω-languages (Corollary 5.3), whereas $XL_{ran,\subseteq}$ and $XL_{inf,\subseteq}$ do (Theorem 3.12). □

Note that the case $(inf,=)$ is left open. It is known that the inclusion d-$XL_{inf,=} \subseteq XL_{inf,=}$ is proper for pushdown automata and Turing machines [7, 8]. A way to obtain a general result would be to give an ω-language not in $\mathscr{B}(\mathscr{F}_\sigma)$ which is included in $XL_{inf,=}$ (under some assumption for the storage type X). In [31] the families of ω-languages accepted by Turing machines are investigated in the Borel hierarchy as well as in the arithmetical hierarchy for ω-languages. It is explained that the family of ω-languages $(inf,=)$-accepted by nondeterministic Turing machines is not included in any family of the Borel hierarchy (cf. the closing remarks of Section 3 in [31]). This indicates that the strictness of the inclusion considered can be proved by topological means.

For some of the specific storages studied in the literature it was observed that the families d-$XL_{\sigma,=}$ are closed under complement, and that the families d-$XL_{\sigma,\subseteq}$ and d-$XL_{\sigma,\sqcap}$ are "complementary", i.e., one contains the complements of the ω-languages of the other. This is due to the fact that deterministic automata are often assumed to be total, i.e., they should have a run on every possible input. Since we do not have this requirement we cannot directly derive such a result for arbitrary X-automata. Instead,

we use the following notion which previously has been quite helpful in complementing the *finitary* languages accepted by deterministic X-automata (cf. [11]).

Definition 5.7. Let $X=(C, C_{\text{in}}, P, F, \mu)$ be a storage type. *X with infinite look-ahead*, denoted by $X_{\omega\text{LA}}$, is the storage type $(C, C_{\text{in}}, P', F, \mu')$, where $P'=P\cup\{inf(\mathscr{A})\mid \mathscr{A}$ is an X-automaton$\}$, with $\mu'(x)=\mu(x)$ for each $x\in P\cup F$, and $\mu'(inf(\mathscr{A}))(c)=true$ if and only if there exists an infinite run of $\mathscr{A}$ on Λ starting in $(q_{\text{in}}, \Lambda, c)$, where q_{in} is $\mathscr{A}$'s initial state.

Note that a deterministic $X_{\omega\text{LA}}$-automaton may use tests $inf(\mathscr{A})$, where $\mathscr{A}$ is *non*deterministic. Similarly, for real-time automata: a real-time $X_{\omega\text{LA}}$-automaton may use tests $inf(\mathscr{A})$, where $\mathscr{A}$ has Λ-transitions.

Lemma 5.8. *Let* $L\subseteq\Sigma^\omega$. (1) *If* $L\in\text{d-}XL_{\sigma,=}$, *then* $\Sigma^\omega-L\in\text{d-}X_{\omega\text{LA}}L_{\sigma,=}$.

(2) *If* $L\in\text{d-}XL_{inf,\subseteq}$, *then* $\Sigma^\omega-L\in\text{d-}X_{\omega\text{LA}}L_{inf,\sqcap}$.

(3) *If* $L\in\text{d-}XL_{inf,\sqcap}$, *then* $\Sigma^\omega-L\in\text{d-}X_{\omega\text{LA}}L_{inf,\subseteq}$.

(4) *Analogous results hold for the corresponding families* $\text{dr-}XL_{\sigma,\rho}$ *and* $\text{dr-}X_{\omega\text{LA}}L_{\sigma,\rho}$.

Proof. Let $\mathscr{A}=(Q, \Sigma, \delta, q_{\text{in}}, c_{\text{in}})$ be a deterministic X-automaton.

(1): The complement with respect to Σ^ω of the ω-language $L_{\sigma,=}(\mathscr{A}, \mathscr{D})$ is equal to $L_{\sigma,=}(\mathscr{A}, 2^Q-\mathscr{D})$, provided for each ω-word in Σ^ω there is a run of $\mathscr{A}$ on this ω-word. Using infinite look-ahead, $\mathscr{A}$ may be transformed in such a way that it satisfies this property. We add a special state q_{fail} to $\mathscr{A}$, together with transitions $(q_{\text{fail}}, a, true, q_{\text{fail}}, \Lambda)$ for $a\in\Sigma$, to which we will lead all "unsuccessful" runs.

There are several possibilities for the behaviour of $\mathscr{A}$ on a given input u to be "unsuccessful".

(a) $\mathscr{A}$ blocks due to an undefined instruction. We can avoid that by testing the instruction as follows. For $\varphi\in F^*$, consider the X-automaton $\mathscr{B}(\varphi)$ consisting of two states q_0 and q_{loop} (of which q_0 is initial), and having two transitions $(q_0, \Lambda, true, q_{\text{loop}}, \varphi)$ and $(q_{\text{loop}}, \Lambda, true, q_{\text{loop}}, \Lambda)$. The X-automaton $\mathscr{B}(\varphi)$ has an infinite run on Λ starting in (q_0, Λ, c) if and only if $\mu(\varphi)(c)$ is defined. Now replace in $\mathscr{A}$ each transition $(q, a, \beta, q', \varphi)$ by the transitions $(q, a, \beta\wedge inf(\mathscr{B}(\varphi)), q', \varphi)$ and $(q, a, \beta\wedge\neg inf(\mathscr{B}(\varphi)), q_{\text{fail}}, \Lambda)$.

(b) $\mathscr{A}$ has an infinite run on a finite prefix of u. We replace in $\mathscr{A}$ each transition $(q, a, \beta, q', \varphi)$ by the transitions $(q, a, \beta\wedge\neg inf(\mathscr{A}(q)), q', \varphi)$ and $(q, a, \beta\wedge inf(\mathscr{A}(q)), q_{\text{fail}}, \Lambda)$, where $\mathscr{A}(q)$ is the X-automaton that equals $\mathscr{A}$, except that its initial state is q.

(c) As a last possibility, $\mathscr{A}$ may block because in some instantaneous description there are no "useful" transitions: the present configuration satisfies none of the tests of the transitions that start in the present state (with a suitable input). To take care of this, we construct new transitions leading to q_{fail} whenever such a situation occurs. For a state q of $\mathscr{A}$ let $\beta_\Lambda(q)$ be the disjunction of all tests in Λ-transitions starting in state q; this means that the automaton can make a Λ-step in some instantaneous

description (q, x, c) if and only if $\mu(\beta_\Lambda(q))(c) = true$. Similarly, we define $\beta_a(q)$ for each letter a in Σ. We add to $\mathscr{A}$ for each state q and each $a \in \Sigma$ the transition $(q, a, \neg\beta_\Lambda(q) \wedge \neg\beta_a(q), q_{\text{fail}}, \Lambda)$.

It is important to note that this construction does not change the (σ, ρ)-accepted ω-language of $\mathscr{A}$ with respect to $\mathscr{D}$, when $(\sigma, \rho) \neq (ran, \sqcap)$; these acceptance types can be used to single out those runs which do not enter the state q_{fail} (or, equivalently, which do not enter q_{fail} from some moment on). This is not true for $(ran, \sqcap)$-acceptance: a run may first enter some "accepting" state and reach q_{fail} afterwards; this leads to an accepting run which originally did not exist.

(2) and (3): For $(inf, \subseteq)$ and $(inf, \sqcap)$-acceptance we may assume that we have a single state set D with respect to which we accept runs (see Lemma 2.7). We have demonstrated above that we may assume that $\mathscr{A}$ has a (unique) run on each ω-word from Σ^ω, but then $\Sigma^\omega - L_{inf, \sqcap}(\mathscr{A}, \{D\})$ is equal to $L_{inf, \subseteq}(\mathscr{A}, \{Q - D\})$.

(4): Note that we have introduced no Λ-transitions in the above construction. Λ-transitions were only used in the look-ahead automaton $\mathscr{B}(\varphi)$, which is allowed. □

Note that if $X_{\omega\text{LA}}$ can be simulated by X (and this holds, e.g., for $X = FS$ and $X = PD$), then Lemma 5.8 shows that d-$XL_{\sigma, =}$ is closed under complement, and that d-$XL_{inf, \subseteq}$ and d-$XL_{inf, \sqcap}$ contain the complements of each others ω-languages. In Section 6 we need this property for a specific storage type.

Remark 5.9. It is perhaps interesting to note that using look-ahead every deterministic automaton $\mathscr{A}$ can be transformed into an equivalent deterministic automaton having finite delay. This is done by adding to each transition starting in a state q, the test $\neg inf(\mathscr{A}(q))$, where $\mathscr{A}(q)$ is the automaton that equals $\mathscr{A}$ except that its initial state is q. Obviously, this implies that d-$XL_{\sigma, \rho} \subseteq$ df-$X_{\omega\text{LA}} L_{\sigma, \rho}$. Using Corollary 4.8, we then reobtain $0^*1 \cdot \{0, 1\}^\omega \notin$ d-$XL_{ran, \subseteq}$ and $(0^*1)^\omega \notin$ d-$XL_{inf, \subseteq}$.

6. A universal storage type

In this section we study a storage type of "maximal power", in the sense that it can be used to simulate any other storage type. We show that most of its families of accepted ω-languages coincide with the families from the Borel hierarchy which were used in Section 5 as topological upper bounds (Lemma 5.2). This illustrates within our framework the strong connections that hold between acceptance types and topological families.

Similar results were obtained by Arnold (in [1]) for the more general framework of transition systems. (They were reobtained in an elegant way in [30] using the relation between deterministic and nondeterministic systems.) Considered in [1] are the acceptance types $(ran, \subseteq)$, $(inf, \subseteq)$, and $(inf, \sqcap)$ – somewhat reformulated to deal with a possibly infinite number of states – for various kinds of transition systems. It should

be intuitively clear that *deterministic*, *finitely branching*, and *countably branching* transition systems (as defined in [1]) correspond, in our framework, closely to automata that are deterministic, have finite delay, or are unrestricted, respectively. Note that the definition of transition system given in [1] does not allow Λ-transitions, whereas in our framework the number of transitions applicable to an *ID* (i.e., the "branching" of an automaton) is bounded by a constant (depending on the automaton).

We give the definition of our maximal storage type U. Intuitively, a U-atomaton has a storage consisting of a one-way write-only tape. The automaton can test, for any finitary language, whether or not the finite word on its tape belongs to the language.

Definition 6.1. The *universal storage type* U equals $(\Gamma^*, \{\Lambda\}, P, F, \mu)$, where Γ is a fixed infinite set of symbols, $P=\{in\,K \mid K\subseteq\Sigma^*$, for a finite $\Sigma\subseteq\Gamma\}$, $F=\{store(x)\mid x\in\Gamma^*\}$, *and, for* $c\in\Gamma^*$, $\mu(in\,K)(c)=true$ iff $c\in K$, and $\mu(store(x))(c)=cx$.

Lemma 6.2. *For every storage type* X, $X\leqslant_r U$.

Proof. Let $X=(C, C_{in}, P, F, \mu)$, let $\mathcal{M}$ be a deterministic X-transducer with initial configuration c_{in}, and let D be a set of states of $\mathcal{M}$. We will construct a deterministic U-transducer $\mathcal{M}'$ (with the same state set as $\mathcal{M}$) such that $T_*(\mathcal{M}, D)=T_*(\mathcal{M}', D)$. The main idea behind this construction is to use the configurations of U to store the sequences of instruction symbols that are performed by $\mathcal{M}$ and to encode, in a suitable language, those sequences that lead to a configuration in which a given test is satisfied.

Let $F_{/\!/}$ be the (finite) subset of F of instruction symbols that are used in $\mathcal{M}$. Without restriction, we may assume that $F_{/\!/}$ is included in Γ, the alphabet of U. Let, for $\varphi\in F^*$, $Def(\varphi)=\{\psi\in F^*_{/\!/}\mid \mu(\psi.\varphi)(c_{in})$ is defined$\}$ and, for $\beta\in BE(P)$, let $True(\beta)=\{\psi\in F^*_{/\!/}\mid c=\mu(\psi)(c_{in})$ is defined and $\mu(\beta)(c)=true\}$. Now $\mathcal{M}'$ is obtained by replacing every transition $(q, a, \beta, q', \varphi, w)$ of $\mathcal{M}$ by the transition $(q, a, in\,True(\beta)\wedge in\,Def(\varphi), q', store(\varphi), w)$.

Clearly, if $\mathcal{M}$ is real-time, then so is $\mathcal{M}'$. □

In particular, we have $U_{\omega LA}\leqslant_r U$ and, consequently, dr-$UL_{\sigma,\rho}$=dr-$U_{\omega LA}L_{\sigma,\rho}$ (Corollary 4.13). This will be useful in the proof of the following result.

Recall that for families $\mathcal{K}$ and $\mathcal{L}$ of ω-languages we use $\mathcal{K}\wedge\mathcal{L}$ to denote $\{K\cap L\mid K\in\mathcal{K}, L\in\mathcal{L}\}$, and $\mathcal{B}(\mathcal{K})$ to denote the Boolean closure of the family $\mathcal{K}$.

Theorem 6.3. (1) dr-$UL_{ran,\subseteq}$ = d-$UL_{ran,\subseteq}=\mathcal{F}$.
(2) dr-$UL_{ran,\sqcap}$ = d-$UL_{ran,\sqcap}=\mathcal{F}\wedge\mathcal{G}$.
(3) dr-$UL_{ran,=}$ = d-$UL_{ran,=}=\mathcal{B}(\mathcal{F})$.
(4) dr-$UL_{inf,\subseteq}$ = d-$UL_{inf,\subseteq}=\mathcal{F}_\sigma$.
(5) dr-$UL_{inf,\sqcap}$ = d-$UL_{inf,\sqcap}=\mathcal{G}_\delta$.
(6) dr-$UL_{inf,=}$ = d-$UL_{inf,=}=\mathcal{B}(\mathcal{F}_\sigma)$.

Proof. The topological upper bounds follow from Lemma 5.2. We prove the inclusion of the six topological families in the respective families of ω-languages accepted by deterministic real-time U-automata.

(1) and (5): Let $K \subseteq \Sigma^*$ and let $\mathscr{A}$ be the U-automaton with transitions $(q_i, a, in\ K, q_1, store(a))$ and $(q_i, a, \neg in\ K, q_0, store(a))$ for $a \in \Sigma$, $i \in \{0, 1\}$, and with initial state q_1. For this automaton we have $L_{inf,\sqcap}(\mathscr{A}, \{\{q_1\}\}) = lim(K)$. This shows that $\mathscr{G}_\delta \subseteq \text{dr-}UL_{inf,\sqcap}$ (see Proposition 1.2).

If, additionally, $K = pref(K)$, then $L_{ran,\subseteq}(\mathscr{A}, \{\{q_1\}\}) = adh(K)$. Since for every language K, $adh(K) = adh(pref(K))$, the assumption $K = pref(K)$ is no loss of generality. This shows that $\mathscr{F} \subseteq \text{dr-}UL_{ran,\subseteq}$.

(2): Let $K, L \subseteq \Sigma^*$, and let $\mathscr{B}$ be the U-automaton with transitions $(q_0, a, \neg in\ L, q_0, store(a))$, $(q_0, a, in\ L, q_1, store(a))$, and $(q_1, a, in\ K, q_1, store(a))$ for $a \in \Sigma$, and with initial state q_0. If $K = pref(K)$, then $L_{ran,\sqcap}(\mathscr{A}, \{\{q_1\}\}) = L.\Sigma^\omega \cap adh(K)$. This shows that $\mathscr{F} \wedge \mathscr{G} \subseteq \text{dr-}UL_{ran,\sqcap}$.

(3): In order to show the inclusion $\mathscr{B}(\mathscr{F}) \subseteq \text{d-}UL_{ran,=}$, we demonstrate that $\mathscr{F} \subseteq \text{dr-}UL_{ran,=}$, and that $\text{dr-}UL_{ran,=}$ is closed under the operations complement and union. Since $\text{dr-}UL_{ran,\subseteq} \subseteq \text{dr-}UL_{ran,=}$ [Theorem 5.5(2)], the inclusion $\mathscr{F} \subseteq \text{dr-}UL_{ran,=}$ follows from (1). The closure of $\text{dr-}UL_{ran,=} = \text{dr-}U_{\omega\text{LA}}L_{ran,=}$ under complement is a consequence of Lemma 5.8. So, it remains to prove the closure of $\text{dr-}UL_{ran,=}$ under union. This is done as follows. Given two deterministic real-time U-automata $\mathscr{A}_1$ and $\mathscr{A}_2$ we construct a deterministic real-time $U \times U$-automaton $\mathscr{A}$ as the product of $\mathscr{A}_1$ and $\mathscr{A}_2$ in an obvious way. If we assume that both $\mathscr{A}_1$ and $\mathscr{A}_2$ have a run on each input word, then we may use Lemma 2.8 to find for each family $\mathscr{D}_i$ a family $\mathscr{D}_i'$ such that $L_{ran,=}(\mathscr{A}_i, \mathscr{D}_i) = L_{ran,=}(\mathscr{A}, \mathscr{D}_i')$. According to the proof of Lemma 5.8, it is no restriction to make this assumption. But then $L_{ran,=}(\mathscr{A}_1, \mathscr{D}_1) \cup L_{ran,=}(\mathscr{A}_2, \mathscr{D}_2) = L_{ran,=}(\mathscr{A}, \mathscr{D}_1' \cup \mathscr{D}_2')$. Since $U \times U \leqslant_r U$ (Lemma 6.2) this proves the closure of $\text{dr-}UL_{ran,=}$ under union (Corollary 4.13).

(4): $\mathscr{F}_\sigma \subseteq \text{dr-}UL_{inf,\subseteq}$ follows by complementation from (5) (see Lemma 5.8).

(6): For the inclusion $\mathscr{B}(\mathscr{F}_\sigma) \subseteq \text{dr-}UL_{inf,=}$ we use an argument analogous to the one in (3). □

Note that these classes are related by the diagram of Fig. 4. Note also that for automata accepting finitary languages U is of no interest: every finitary language can be accepted by a deterministic U-automaton.

According to the previous result, in the deterministic case the maximal power of U can be expressed as a topological family, depending on the acceptance criterion. Also the deterministic U-transductions are of topological significance (cf. [33]), as shown in the next result.

Theorem 6.4. (1) $\text{dr-}UT = \text{d-}UT$ *equals the family of continuous functions with domain in* $\mathscr{G}_\delta$.

(2) $\text{dr-}UT_\omega = \text{d-}UT_\omega$ *equals the family of continuous functions with domain in* $\mathscr{F}$.

Proof. Recall that the function $f: \Sigma^\omega \to \Delta^\omega$ is continuous in a word u if for each $m \in \mathbb{N}$ there exists $n \in \mathbb{N}$ such that $f(u[n].\Sigma^\omega) \subseteq f(u)[m].\Delta^\omega$.

(a) Clearly, every deterministic transducer defines a continuous function: if $(u, v) \in T(\mathcal{M})$ for some deterministic transducer $\mathcal{M}$, and $\mathcal{M}$ outputs the first m symbols of v on the first n symbols of u, then $T(\mathcal{M})(u[n].\Sigma^\omega) \subseteq v[m].\Delta^\omega$, where Σ and Δ are the input alphabet and output alphabet of $\mathcal{M}$.

Regarding the domain of transductions, we observe that it is straightforward to extend Lemma 2.13 to deterministic transductions. Hence, $dom(\text{d-}UT) = \text{d-}UL_{inf, \sqcap} = \mathcal{G}_\delta$, and $dom(\text{d-}UT_\omega) = \text{d-}UL_{ran, \subseteq} = \mathcal{F}$.

(b) We now have to show that every continuous function (with a suitable domain) can be implemented as a deterministic real-time (ω-preserving) U-automaton. We will do this in an indirect way by using the storage type $FUNC(f, K)$, which allows one to simulate functions in a simple way, rather than the storage type U. The result then follows since by Lemmas 6.2 and 4.11, $\text{dr-}FUNC(f, K)T_\omega \subseteq \text{dr-}UT_\omega$, and $\text{dr-}FUNC(f, K)T \subseteq \text{dr-}UT$.

For a given function $f: \Sigma^\omega \to \Delta^\omega$ and a language $K \subseteq \Sigma^*$ the storage type $FUNC(f, K)$ is given by $(\Sigma^* \times \Delta^*, \{(\Lambda, \Lambda)\}, P, F, \mu)$, where P contains the predicate symbols $next(b)$, for every $b \in \Sigma$, $nonext$, and $in\,K$, and F contains the instruction symbols $store_1(a)$, for every $a \in \Sigma$, and $store_2(b)$, for every $b \in \Delta$. The meaning of these symbols is given by

$$\mu(next(b))(x, y) = true \text{ iff } f(x.\Sigma^\omega) \subseteq y.b.\Delta^\omega,$$

$$\mu(nonext)(x, y) = true \text{ iff there is no } b \in \Delta \text{ such that } f(x.\Sigma^\omega) \subseteq y.b.\Delta^\omega,$$

$$\mu(in\,K)(x, y) = true \text{ iff } x \in K,$$

$$\mu(store_1(a))(x, y) = (x.a, y)$$

and, similarly,

$$\mu(store_2(b))(x, y) = (x, y.b).$$

Let $\mathcal{M}$ be the deterministic real-time one-state $FUNC(f, K)$-transducer with the transitions $(q, a, \neg in\,K, q, store_1(a), \Lambda)$, $(q, a, in\,K \wedge next(b), q, store_1(a)store_2(b), b)$, and $(q, a, in\,K \wedge nonext, q, store_1(a), \Lambda)$, for $a \in \Sigma$ and $b \in \Delta$.

(b.1) Note that $\mathcal{M}$ can only output a letter if the prefix of the input which has been read belongs to K. Hence, the domain of $T(\mathcal{M})$ is included in $lim(K)$. On the other hand, if f is a continuous function with domain $lim(K)$, then the continuity will guarantee that infinitely often for some appropriate $b \in \Delta$ the test $next(b)$ is satisfied. Hence, $\mathcal{M}$ then realizes the function f: $T(\mathcal{M}) = \{(u, v) \mid u \in lim(K), f(u) = v\}$. This proves (1) of the theorem.

(b.2) Assume now that f is a continuous function with domain $adh(K)$ in $\mathcal{F}$, and assume that the language K is prefix closed. We transform $\mathcal{M}$ into an ω-preserving

transducer by omitting the transitions $(q, a, \neg \text{in } K, q, store_1(a), \Lambda)$, $a \in \Sigma$. Again $\mathcal{M}$ realizes f: $T(\mathcal{M}) = \{(u, v) | u \in adh(K), f(u) = v\}$. This proves (2) of the theorem. □

We now turn to nondeterministic U-automata.

Theorem 6.5. *$UL_{\sigma,\rho}$ equals the family of continuous images of $\mathcal{G}_\delta$-sets.*

Proof. Since, according to Theorem 3.11 and Lemma 6.2, all families $UL_{\sigma,\rho}$ are equal, it suffices to consider one of them. On the one hand, by Lemma 3.8, $UL_{inf,\sqcap} = HOM(\text{d-}UL_{inf,\sqcap}) \subseteq \text{dr-}FST(\text{d-}UL_{inf,\sqcap})$. This implies that each set in $UL_{inf,\sqcap}$ is the continuous image of the intersection of two $\mathcal{G}_\delta$-sets: the domain of the transducer [see Theorem 6.4(1)] and the d-$UL_{inf,\sqcap}$ set (Theorem 6.3) which again is a $\mathcal{G}_\delta$-set.

On the other hand, again by Theorem 6.4(1), the continuous images of $\mathcal{G}_\delta$-sets are exactly the ranges of deterministic real-time U-transductions. We have the inclusions $ran(\text{dr-}UT) \subseteq ran(UT) = UL_{inf,\sqcap}$ [cf. Lemma 2.13(1)]. □

Continuous functions on ω-languages were studied in [33], where they were called sequential mappings. The continuous images of $\mathcal{G}_\delta$-sets are known under the name analytic sets (or Souslin sets, sets of first projective class) in the literature. They are equal to the continuous images (or projections) of the Borel sets.

Since $Q \times U \leqslant_r U$, the relations between the families r-$UL_{\sigma,\rho}$ and f-$UL_{\sigma,\rho}$ are already given in Theorem 4.17. Now we are able to give the exact topological characterizations of these families (cf. Lemma 4.7).

Theorem 6.6. (1) $\text{r-}UL_{ran,\subseteq} = \text{f-}UL_{ran,\subseteq} = \mathcal{F}$.

(2) $\text{r-}UL_{inf,\subseteq} = \text{f-}UL_{inf,\subseteq} = \mathcal{F}_\sigma$.

(3) $\text{r-}UL_{inf,\sqcap} = \text{f-}UL_{inf,\sqcap} = UL_{inf,\sqcap}$.

Proof. The topological upper bounds f-$UL_{ran,\subseteq} \subseteq \mathcal{F}$ and f-$UL_{inf,\subseteq} \subseteq \mathcal{F}_\sigma$ follow from Lemma 4.7. The converse inclusions $\mathcal{F} \subseteq \text{r-}UL_{ran,\subseteq}$ and $\mathcal{F}_\sigma \subseteq \text{r-}UL_{inf,\subseteq}$ are shown in Theorem 6.3.

The equalities in (3) follow from Theorem 4.17 (and Lemma 6.2). □

The relations between acceptance types and topological families were considered in this paper at a rather elementary level, as a simple technical tool to provide us with examples to prove the strictness of some inclusions. A deeper study of the ω-languages accepted by Turing machines, and their relation to the arithmetical hierarchy from recursion theory and the topological Borel hierarchy is presented in [31]. In [37] a common framework underlying these two hierarchies is presented, with some explicit comments on the technical differences between the classical definitions and their adaptation to language theory (where finite alphabets and ω-words take the place of natural numbers and number-theoretic functions). It also contains variants of these hierarchies based on regular ω-languages.

7. Logical acceptance criteria

In this paper we have studied, within a common framework, the acceptance of ω-languages for several types of automata. We have illustrated our methods by investigating both unrestricted automata, as well as some restrictions like real time and determinism. We did not succeed in deriving all related results from the literature using our general approach, but some interesting phenomena (such as the strictness of the inclusion diagrams for real-time and deterministic automata) could be generalized to X-automata.

The acceptance types (σ, ρ) we have used are the six conditions one usually finds in the literature. The reader may wonder whether this choice is not too restrictive: certainly, there should be other, natural, acceptance conditions that cannot be expressed as some property of the range or the infinity set of the state sequence of a run. If this were true, then a broad framework for studying the acceptance of ω-languages should not only allow arbitrary storage types but also a general notion of acceptance.

If we restrict the acceptance criterion to be a property of the state sequence of a run (including the contents of the storage will change the theory radically), it is natural to require that this property can be expressed in some well-defined formal language.

A well-known language for specifying properties of infinite sequences (i.e., ω-words over some alphabet) is Büchi's *sequential calculus*, a monadic second-order logic [3]. This logic is powerful enough to express each of the (σ, ρ)-acceptance types (even as *first*-order formulas). We will show in this section the converse of this fact: for X-automata all acceptance criteria definable in the sequential calculus will give ω-languages inside $XL_{inf,=}$, i.e., $(inf, =)$-acceptance is as powerful as monadic second-order acceptance. This generalizes one of the results from [19], stating that first-order acceptance is as powerful as $(inf, =)$-acceptance for finite-state automata, in two respects: we consider second-order formulas for X-automata rather than just first-order formulas for finite-state automata.

For a fixed alphabet A, we will denote Büchi's sequential calculus here by MSO_A; its formulas will be referred to as *A-formulas*.

MSO_A contains variables $i, j, k, \ldots$ (ranging over $\mathbb{N}$) and set-variables $U, V, \ldots$ (ranging over $2^{\mathbb{N}}$), used to indicate positions, and sets of positions, respectively, in an ω-word. The *terms* of MSO_A are constructed from the constant 0 and the variables $i, j, \ldots$ by applying the successor-function $+1$.

The *atomic formulas* of MSO_A are of one of the forms $t_1 < t_2$, $t_1 \in U$, or $P_a(t_1)$, where t_1 and t_2 are terms, U is a set variable, and $a \in A$. Here $<$ and $\in$ have their usual meaning; $P_a(m)$ means that the mth letter of the ω-word equals a. From these atomic formulas we construct the A-formulas in the usual way using the connectives $\neg$, $\vee$, $\wedge$, $\rightarrow$, and the quantifiers $\exists$ and $\forall$ (for both types of variables).

First of all (as in the work of Büchi [3], see also the exposition in [36]) such a formula can be used directly to specify a property of ω-words and, consequently, to

define the ω-language consisting of the ω-words that satisfy the formula. On the other hand (as is done in [19]), the formula may also be used in an indirect fashion to define ω-languages by specifying an acceptance condition for an automaton, i.e., by specifying accepting state sequences of runs.

We will give the corresponding formal definitions.

Let Σ be an alphabet. For a closed Σ-formula φ of MSO_Σ, the *ω-language defined by* φ equals $L(\varphi)=\{u\in\Sigma^\omega \mid u \text{ satisfies } \varphi\}$. We use $MSOL$ to denote the family of these mso-definable ω-languages.

Given an X-automaton $\mathscr{A}$ with state set Q and input alphabet Σ, and a closed Q-formula φ, the *ω-language φ-accepted by $\mathscr{A}$*, denoted by $L(\mathscr{A}, \varphi)$, is $\{u\in\Sigma^\omega \mid$ there is a run of $\mathscr{A}$ on u of which the state sequence satisfies $\varphi\}$. For any collection Φ of monadic second-order formulas, XL_Φ is the corresponding family of ω-languages that can be accepted by X-automata using monadic second-order formulas from Φ; in particular, XL_{mso} and XL_{fo} are the families where all mso formulas are allowed, respectively, where only first-order formulas are allowed (i.e., the ones not involving set variables).

As an example, the ω-language $(0^*1)^\omega$ is defined by the $\{0,1\}$-formula $\forall i\exists j\,(j>i\wedge P_1(j))$. All the (σ,ρ)-acceptance types are mso-expressible; e.g., if $\mathscr{A}$ is an X-automaton with state set Q, and $\mathscr{D}$ is a family of state sets for $\mathscr{A}$, then $L_{inf,=}(\mathscr{A},\mathscr{D})=L(\mathscr{A},\varphi)$, where φ is the formula

$$\bigvee_{D\in\mathscr{D}}\bigwedge_{q\in Q}(q\in D\leftrightarrow\forall i\exists j(j>i\wedge P_q(j))).$$

From the results of Büchi and McNaughton [3, 25] we know that the family of mso-definable ω-languages coincides with the family of regular ω-languages. On the other hand, when considering logical formulas to specify acceptance conditions, one of the results obtained in [19] shows that also the ω-languages accepted by deterministic finite-state automata using a first-order definable acceptance condition are exactly the regular ω-languages.

Proposition 7.1. (1) $\mathrm{d}\text{-}FSL_{inf,=}=FSL_{inf,=}=MSOL$.

(2) $\mathrm{d}\text{-}FSL_{inf,=}=\mathrm{d}\text{-}FSL_{\mathrm{fo}}$.

As already stated in the introduction above we will extend the result of [19] to monadic second-order acceptance for arbitrary storage types. For the storage type FS the equality $FSL_{inf,=}=FSL_{\mathrm{mso}}$ can be shown directly with a simple variation of the ideas used by Büchi and in [19]. We will prove the result for arbitrary storage types by applying Büchi's characterization and the decomposition technique we have used before.

Theorem 7.2. (1) $XL_{inf,=}=XL_{\mathrm{mso}}$.

(2) $\mathrm{d}\text{-}XL_{inf,=}=\mathrm{d}\text{-}XL_{\mathrm{mso}}$.

Proof. (1): The equality is shown using a series of inclusions.

(i) $XL_{inf,=} \subseteq XL_{\mathrm{mso}}$. As we have seen, the property "for one of the sets $D \in \mathscr{D}$ each state occurs infinitely often if and only if it belongs to D" is monadic second-order (and even first-order) expressible.

(ii) $XL_{\mathrm{mso}} \subseteq XT_\omega^{-1}(MSOL)$. As in the proof of Lemma 3.1, an X-automaton $\mathscr{A}$ (with state set Q) can be transformed into an X-transducer $\mathscr{M}$ with the same behaviour as $\mathscr{A}$ except that it outputs its state in each transition. Any Q-formula acting as acceptance condition for $\mathscr{A}$ can now be tested on the output of $\mathscr{M}$:

$$\begin{aligned} L(\mathscr{A}, \varphi) &= \{u \in \Sigma^\omega \mid \text{there is a run of } \mathscr{A} \text{ on } u \text{ of which the state sequence satisfies } \varphi\} \\ &= \{u \in \Sigma^\omega \mid \text{there is a run of } \mathscr{M} \text{ on } u \text{ with output satisfying } \varphi\} \\ &= T^{-1}(\mathscr{M})(L(\varphi)). \end{aligned}$$

(iii) $XT_\omega^{-1}(MSOL) \subseteq XT_\omega^{-1}(FSL_{inf,=})$. This is due to Büchi's characterization [Proposition 7.1(1)].

(iv) $XT_\omega^{-1}(FSL_{inf,=}) \subseteq XL_{inf,=}$, by Theorem 3.3.

(2): The proof of the deterministic case is analogous. □

Corollary 7.3. $FSL_{\mathrm{mso}} = MSOL$.

Proof. Take $X = FS$ in Theorem 7.2 and combine with Proposition 7.1(1). □

Also some of the other characterizations given in [19] can be extended to X-automata. Let Π_2 be the subset of closed first-order formulas of the form $\forall i_i \ldots \forall i_m \exists j_1 \ldots \exists j_n \psi(i_1, \ldots i_m, j_1, \ldots j_n)$, where $\psi(\cdots)$ is a formula without quantifiers, and let d-XL_{Π_2} and $\Pi_2 L$ be the corresponding subfamilies of d-XL_{mso}, and $MSOL$, respectively. Then it is shown in [19] that d-$FSL_{inf,\sqcap} =$ d-FSL_{Π_2}.

Using this equality, the corresponding equality d-$XL_{inf,\sqcap} =$ d-XL_{Π_2} for X-automata can be obtained using a series of inclusions like those given in the proof of Theorem 7.2. One has d-$XL_{inf,\sqcap} \subseteq$ d-$XL_{\Pi_2} \subseteq$ d-$XT_\omega^{-1}(\Pi_2 L) \subseteq$ d-XT_ω^{-1}(d-$FSL_{inf,\sqcap}$) $\subseteq$ d-$XL_{inf,\sqcap}$, where the inclusions are shown as before, except that we need a new argument for the inclusion $\Pi_2 L \subseteq$ d-$FSL_{inf,\sqcap}$ which replaces Büchi's characterization in this proof. By the result of [19] it suffices to prove the inclusion $\Pi_2 L \subseteq$ d-FSL_{Π_2}.

Let φ be a Σ-formula in Π_2 for some alphabet Σ. We will construct a FS-automaton with a Π_2-acceptance condition accepting $L(\varphi)$. Similar to the proof of Lemma 3.1, consider the deterministic finite-state automaton $\mathscr{B} = (\Sigma \cup \{q^0\}, \Sigma, \delta_{\mathscr{B}}, q^0, c0)$ with $q^0 \notin \Sigma$ and $\delta_{\mathscr{B}} = \{(\sigma', \sigma, \mathit{true}, \sigma, \Lambda) \mid \sigma' \in \Sigma \cup \{q^0\}, \sigma \in \Sigma\}$. One easily sees that for $u \in \Sigma^\omega$, the state sequence of the corresponding run r of $\mathscr{B}$ on u equals $q^0 u$ (i.e., it equals u except for the initial q^0). Let φ' be the formula that one obtains from φ by changing each predicate $P_\sigma(t)$ into $P_\sigma(t+1)$. Then u satisfies φ if and only if $q^0 u$ satisfies φ'. This implies that $L(\varphi) = L(\mathscr{B}, \varphi')$.

Similarly, for the subclass Π_1 of closed first-order formulas of the form $\forall i_1 \ldots \forall i_m \psi(i_1, \ldots, i_m)$, the characterization d-$FSL_{ran,\subseteq}$ = d-FSL_{Π_1} from [19] leads to the same result for X-automata.

It would be interesting to develop a theory of X-automata with a general notion of acceptance. As suggested above one could define the notion of acceptance criterion to be a set Φ of *MSO* formulas, satisfying certain natural conditions. These conditions should be taken in such a way that one could prove, e.g., the analogue of Theorem 3.3.

Acknowledgment

We thank Dr. Ludwig Staiger and the referee for many useful suggestions, and Prof. Wolfgang Thomas for several motivating discussions.

References

[1] A. Arnold, Topological characterizations of infinite behaviours of transition systems, in: J. Diaz, ed., *Proc. ICALP 1983*, Lecture Notes in Computer Science, Vol. 154 (Springer, Berlin, 1983) 28–38.

[2] L. Boasson and M. Nivat, Adherences of languages, *J. Comput. System Sci.* **20** (1980) 285–309.

[3] J.R. Büchi, On a decision method in restricted second order arithmetic, in: *Proc. Internat. Congr. on Logic, Methodology and Philosophy of Science 1960* (Stanford University Press, Stanford, CA, 1962) 1–11.

[4] H. Carstensen, Fairness in deadlockfree Petri nets with the finite delay property, in: *Proc. 5th European Workshop on Applications and Theory of Petri Nets*, Aarhus (1984) 234–253.

[5] H. Carstensen, Infinite behaviour of deterministic Petri nets, in: M.P. Chytil, L. Janiga and V. Koubek, eds., *Proc. MFCS 1988*, Lecture Notes in Computer Science, Vol. 324 (Springer, Berlin, 1988) 210–219.

[6] R.S. Cohen and A.Y. Gold, Theory of ω-languages. I: characterizations of ω-context-free languages, II: a study of various models of ω-type generation and recognition, *J. Comput. System Sci.* **15** (1977) 169–208.

[7] R.S. Cohen and A.Y. Gold, ω-Computations on deterministic pushdown machines, *J. Comput. System Sci.* **16** (1978) 275–300.

[8] R.S. Cohen and A.Y. Gold, ω-Computations on Turing machines, *Theoret. Comput. Sci.* **6** (1978) 1–23.

[9] S. Eilenberg, *Automata, Languages and Machines* (Academic Press, New York and London, 1974) Chapter XIV.

[10] J. Engelfriet and H.J. Hoogeboom, Automata with storage on infinite words, in: G. Ausiello, M. Dezani-Ciancaglini and S. Ronchi Della Rocca, eds., *Proc. ICALP 1989*, Lecture Notes in Computer Science, Vol. 372 (Springer, Berlin, 1989) 389–303.

[11] J. Engelfriet and H. Vogler, Look-ahead on pushdowns, *Inform. and Comput.* **73** (1987) 245–279.

[12] S. Ginsburg, *Algebraic and Automata-Theoretic Properties of Formal Languages* (North-Holland/American Elsevier, Amsterdam/New York, 1975).

[13] S. Ginsburg and S.A. Greibach, Abstract families of languages, in: *Studies in Abstract Families of Languages, Mem. Amer. Math. Soc.* **87** (1969) 1–32.

[14] S.A. Greibach, Remarks on blind and partially blind one-way multicounter machines, *Theoret. Comput. Sci.* **7** (1978) 311–324.

[15] H.J. Hoogeboom and G. Rozenberg, Infinitary languages – basic theory and applications to concurrent systems, in: J.W. de Bakker, W.P. de Roever and G. Rozenberg, eds., *Current trends in Concurrency*, Lecture Notes in Computer Science, Vol. 224 (Springer, Berlin, 1986) 266–342.

[16] J.E. Hopcroft and J.D. Ullman, An approach to a unified theory of automata, *The Bell System Technical Journal* **XLVI** (1967) 1793–1829.
[17] R. Hossley, Finite tree automata and ω-automata, MAC Tech. Report 102 MIT, 1972.
[18] M. Jantzen, On the hierarchy of Petri net languages, *RAIRO Inform. Théor. Appl.* **13** (1979) 19–30.
[19] K. Kobayashi, M. Takahashi and H. Yamasaki, Characterization of ω-regular languages by first order formulas, *Theoret. Comput. Sci.* **28** (1984) 315–327.
[20] L.H. Landweber, Decision problems for ω-automata, *Math. Systems Theory* **3** (1969) 376–384.
[21] M. Latteux and E. Timmerman, Two characterizations of rational adherences, *Theoret. Comput. Sci.* **46** (1986) 101–106.
[22] R. Lindner and L. Staiger, *Algebraische Codierungstheorie – Theorie der sequentiellen Codierungen* (Akademie, Berlin, 1977).
[23] M. Linna, On ω-sets associated with context-free languages, *Inform. and Control* **31** (1976) 272–293.
[24] M. Linna, A decidability result for deterministic ω-context-free languages, *Theoret. Comput. Sci.* **4** (1977) 83–98.
[25] R. McNaughton, Testing and generating infinite sequences by a finite automaton, *Inform. and Control* **9** (1966) 521–530.
[26] D.E. Muller, Infinite sequences and finite machines, in: *AIEE Proc. 4th Ann. Symp. Switch. Circuit Theory and Logical Design* (1963) 3–16.
[27] D. Scott, Some definitional suggestions for automata theory, *J. Comput. System Sci.* **1** (1967) 187–212.
[28] L. Staiger, Empty-storage-acceptance of ω-languages, in: M. Karpiński, ed., *Proc. FCT 77*, Lecture Notes in Computer Science, Vol. 56 (Springer, Berlin, 1977) 516–521.
[29] L. Staiger, Finite-state ω-languages, *J. Comput. System Sci.* **27** (1983) 434–448.
[30] L. Staiger, Projection lemmas for ω-languages, *Theoret. Comput. Sci.* **32** (1984) 331–337.
[31] L. Staiger, Hierarchies of recursive ω-languages, *Elektron. Inform. verarb. Kybern. EIK* **22** (1986) 219–241.
[32] L. Staiger, Research in the theory of ω-languages, *J. Inform. Process. Cybern. EIK* **23** (1987) 415–439.
[33] L. Staiger, Sequential mappings of ω-languages, *RAIRO Inform. Théor. Appl.* **21** (1987) 147–173.
[34] L. Staiger and K. Wagner, Automatentheoretische und automatenfreie Characterisierungen topologischer Klassen regulaerer Folgenmengen, *Elektron. Inform. verarb. Kybern. EIK* **10** (1974) 379–392.
[35] L. Staiger and K. Wagner, Rekursive Folgenmengen I, *Z. Math. Logik Grundlag. Math.* **24** (1978) 523–538.
[36] W. Thomas, Automata on infinite objects, in: J. van Leeuwen, ed., *Handbook of Theoretical Computer Science, Vol. B* (Elsevier, Amsterdam, 1990) 133–191.
[37] W. Thomas, Automata and quantifier hierarchies, Aachener Informatik-Berichte 88-23, RWTH Aachen, FRG, 1988.
[38] R. Valk, Infinite behaviour of Petri nets, *Theoret. Comput. Sci.* **25** (1983) 311–341.
[39] K. Wagner, On ω-regular sets, *Inform. and Control* **43** (1979) 123–177.
[40] K. Wagner and L. Staiger, Recursive ω-languages, in: M. Karpiński, ed., *Proc. FCT 77*, Lecture Notes in Computer Science, Vol. 56 (Springer, Berlin, 1977) 532–537; see the full paper [35].