Lectures on the modal μ-calculus

Yde Venema*

Abstract

These notes give an introduction to the theory of the modal μ-calculus.

*Institute for Logic, Language and Computation, University of Amsterdam, Science Park 107, NL–1098XG Amsterdam. E-mail: `y.venema@uva.nl`.

Contents

4 Stream automata and logics for linear time

As we already mentioned in the introduction in the theory of the modal μ-calculus and other fixpoint logics a fundamental role is played by automata. As we will see further on, these devices provide a very natural generalization to the notion of a formula. This chapter gives an introduction to the theory of automata operating on (potentially infinite) objects. Whereas in later chapters we will meet various kinds of automata for classifying trees and general transition systems, here we confine our attention to the devices that operate on *streams* or infinite words, these being the simplest nontrivial examples of infinite behavior.

Convention 4.1 Throughout this chapter (and the next), we will be dealing with some finite *alphabet* C. Generic elements of C may be denoted as $c, d, c_0, c_1, \ldots$, but often it will be convenient to think of C as a set of *colors*. In this case we will denote the elements of C with lower case roman letters that are mnemonic of the most familiar corresponding color ('b' for *blue*, 'g' for *green*, etcetera).

Definition 4.2 Given an alphabet C, a *C-stream* is just an infinite C-sequence, that is, a map $\gamma : \omega \to C$ from the natural numbers to C (see Appendix A). C-streams will also be called *infinite words* or *ω-words* over C. Sets of C-streams are called *stream languages* or *ω-languages* over C. $\lhd$

Remark 4.3 This definition is consistent with the terminology we introduced in Chapter 1. There we defined a *$\wp(\mathsf{P})$-stream* or *stream model for* P to be a Kripke model of the form $\mathbb{S} = \langle \omega, V, \mathit{Succ} \rangle$, where Succ is the standard successor relation on the set ω of natural numbers, and $V : \mathsf{P} \to \wp(\omega)$ is a valuation. If we represent V coalgebraically as a map $\sigma_V : \omega \to \wp(\mathsf{P})$ (cf. Remark 1.3), then in the terminology of Definition 4.2, $\mathbb{S}$ is indeed a *$\wp(\mathsf{P})$-stream.* $\lhd$

4.1 Deterministic stream automata

We start with the most general definition of a deterministic stream automaton.

Definition 4.4 Given an *alphabet* C, a *deterministic C-automaton* is a quadruple $\mathbb{A} = \langle A, \delta, \mathit{Acc}, a_I \rangle$, where A is a finite set, $a_I \in A$ is the *initial state* of $\mathbb{A}$, $\delta : A \times C \to A$ its *transition function*, and $\mathit{Acc} \subseteq A^\omega$ its *acceptance condition.* The pair $\langle A, \delta \rangle$ is called the *transition diagram* of $\mathbb{A}$.

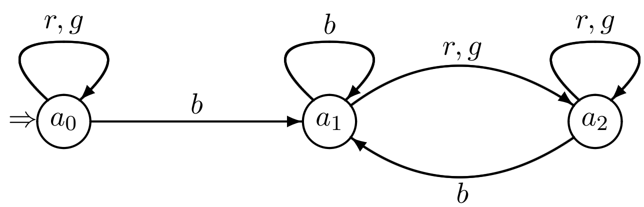
Given an automaton $\mathbb{A} = \langle A, \delta, \mathit{Acc}, a_I \rangle$, we may extend the map $\delta : A \times C \to A$ to a map $\widehat{\delta} : A \times C^* \to A$ by putting

$$\begin{array}{lll} \widehat{\delta}(a, \varepsilon) & := & a \\ \widehat{\delta}(a, uc) & := & \delta(\widehat{\delta}(a, u), c). \end{array}$$

We will write $a \xrightarrow{c} a'$ if $a' = \delta(a, c)$, and $a \overset{w}{\twoheadrightarrow} a'$ if $a' = \widehat{\delta}(a, w)$. In words, $a \overset{w}{\twoheadrightarrow} a'$ if there is a w-labelled path from a to a'.

$\lhd$

Example 4.5 The transition diagram and initial state of a deterministic automaton can nicely be represented graphically, as in the picture below, where $C = \{b, r, g\}$:



$\triangleleft$

An automaton comes to life if we supply it with input, in the form of a stream over its alphabet: It will *process* this stream, as follows. Starting from the initial state a_I, the automaton will step by step pass through the stream, jumping from one state to another as prescribed by the transition function.

Example 4.6 Let $\mathbb{A}_0$ be any automaton with transition diagram and initial state as given above, and suppose that we give this device as input the stream $\alpha = brgbrgbrgbrgbrgb \cdots$. Then we find that $\mathbb{A}_0$ will make an infinite series of transitions, determined by α:

$$a_0 \xrightarrow{b} a_1 \xrightarrow{r} a_2 \xrightarrow{g} a_2 \xrightarrow{b} a_1 \cdots$$

Thus the machine passes through an infinite sequence of states:

$$\rho = a_0 a_1 a_2 a_2 a_1 a_2 a_2 a_1 a_2 a_2 \ldots$$

This sequence is called the *run* of the automaton on the word α — a run of $\mathbb{A}$ is thus an A-stream.

For a second example, on the word $\alpha' = brbgbrgrgrgrgrgr \cdots$ the run of the automaton $\mathbb{A}_0$ looks as follows:

$$a_0 \xrightarrow{b} a_1 \xrightarrow{r} a_2 \xrightarrow{b} a_1 \xrightarrow{g} a_2 \xrightarrow{b} a_1 \xrightarrow{r} a_2 \xrightarrow{g} a_2 \xrightarrow{r} a_2 \xrightarrow{g} \cdots$$

we see that from the sixth step onwards, the machine device remains circling in its state a_2: $\cdots\ a_2 \xrightarrow{r} a_2 \xrightarrow{g} a_2 \xrightarrow{r} \cdots$. $\triangleleft$

Definition 4.7 The *run* of an automaton $\mathbb{A} = \langle A, \delta, Acc, a_I \rangle$ on a C-stream $\gamma = c_0 c_1 c_2 \ldots$ is the infinite A-sequence

$$\rho = a_0 a_1 a_2 \ldots$$

such that $a_0 = a_I$ and $a_i \xrightarrow{c_i} a_{i+1}$ for every $i \in \omega$. $\triangleleft$

Generally, whether or not an automaton *accepts* an infinite word, depends on the existence of a successful run — note that in the present deterministic setting, this run is unique. In order to determine which runs are successful, we need the acceptance condition.

Definition 4.8 A run $\rho \in A^\omega$ of an automaton $\mathbb{A} = \langle A, \delta, Acc, a_I \rangle$ is *successful* with respect to an acceptance condition Acc if $\rho \in Acc$.

An C-automaton $\mathbb{A} = \langle A, \delta, Acc, a_I \rangle$ *accepts* a C-stream γ if the run of $\mathbb{A}$ on γ is successful. The *ω-language* $L_\omega(\mathbb{A})$ associated with $\mathbb{A}$ is defined as the set of streams that are accepted by $\mathbb{A}$. Two automata are called *equivalent* if they accept the same streams. $\lhd$

A natural requirement on the acceptance condition is that it only depends on a bounded amount of information about the run.

Remark 4.9 In the case of automata running on *finite words*, there is a very simple and natural acceptance criterion. The point is that runs on finite words are themselves finite too. For instance, suppose that in Example 4.6 we consider the run on the finite word *brgb*:

$$a_0 \xrightarrow{b} a_1 \xrightarrow{r} a_2 \xrightarrow{g} a_2 \xrightarrow{b} a_1.$$

Then this runs *ends* in the state a_1. In this context, a natural criterion for the acceptance of the word *abca* by the automaton is to make it dependent on the membership of this final state a_1 in a designated set $F \subseteq A$ of *accepting* states.

A structure of the form $\mathbb{A} = \langle A, \delta, F, a_I \rangle$ with $F \subseteq A$ may be called a *finite word automaton*, and we say that such a structure *accepts* a finite word w if the unique state a such that $a_I \overset{w}{\twoheadrightarrow} a$ belongs to F. The *language* $L(\mathbb{A})$ is defined as the set of all finite words accepted by $\mathbb{A}$. $\lhd$

4.2 Acceptance conditions

For runs on infinite words, a natural acceptance criterion would involve the collection of states that occur infinitely often in the run.

Definition 4.10 Let $\alpha : \omega \to A$ be a stream over some finite set A. Given an element $a \in A$, we define the *frequency* of a in α as $\#_a(\alpha) := |\{n \in \omega \mid \alpha(n) = a\}|$. Based on this, we set $Occ(\alpha) := \{a \in A \mid \#_a(\alpha) > 0\}$ and $Inf(\alpha) := \{a \in A \mid \#_a(\alpha) = \omega\}$ $\lhd$

In words, $Occ(\alpha)$ and $Inf(\alpha)$ denote the set of elements of A that occur in α at least once and infinitely often, respectively.

Definition 4.11 Given a transition diagram $\langle A, \delta \rangle$, we define the following types of acceptance conditions:

- A *Muller* condition is given as a collection $\mathcal{M} \subseteq \wp(A)$ of subsets of A. The corresponding acceptance condition is defined as

$$Acc_{\mathcal{M}} := \{\alpha \in A^\omega \mid Inf(\alpha) \in \mathcal{M}\}.$$

- A *Büchi* condition is given as a subset $F \subseteq A$. The corresponding acceptance condition is defined as

$$Acc_F := \{\alpha \in A^\omega \mid Inf(\alpha) \cap F \neq \varnothing\}.$$

- A *parity condition* is given as a map $\Omega : A \to \omega$. The corresponding acceptance condition is defined as

$$Acc_\Omega := \{\alpha \in A^\omega \mid \max\{\Omega(a) \mid a \in \mathit{Inf}(\alpha)\} \text{ is even } \}.$$

Automata with these acceptance conditions are called *Muller*, *Büchi* and *parity automata*, respectively. ◁

Of these three types of acceptance conditions, the Muller condition perhaps is the most natural. It exactly and directly specifies the subsets of A that are admissible as the set $\mathit{Inf}(\rho)$ of a successful run. The Büchi condition is also fairly intuitive: an automaton with Büchi condition F accepts a stream α if the run on α passes through some state in F infinitely often. This makes Büchi automata the natural analog of the automata that operate on *finite* words, see Remark 4.9.

The parity condition may be slightly more difficult to understand. The idea is to give each state a of $\mathbb{A}$ a weight $\Omega(a) \in \omega$. Then any infinite A-sequence $\alpha = a_0a_1a_2\ldots$ induces an infinite sequence $\Omega(a_0)\Omega(a_1)\ldots$ of natural numbers. Since the range of Ω is finite this means that there is a *largest* natural number N_α occurring infinitely often in this sequence, $N_\alpha := \max\{\Omega(a) \mid a \in \mathit{Inf}(\alpha)\}$. Now, a parity automaton accepts an infinite word iff the number N_ρ of the associated run ρ is *even*.

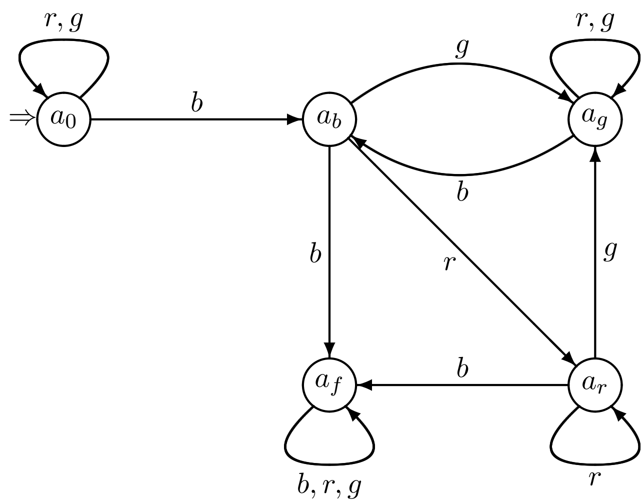
At first sight, this condition will seem rather contrived and artificial. Nevertheless, for a number of reasons the parity automaton is destined to play the leading role in these notes. Most importantly, the distinction between even and odd parities directly corresponds to that between least and greatest fixpoint operators, so that parity automata are the more direct automata-theoretic counterparts of fixpoint formulas. An additional theoretic motivation to use parity automata is that their associated acceptance games have some very nice game-theoretical properties, as we will see further on.

Let us now first discuss some examples of automata with these three acceptance conditions.

Example 4.12 Suppose that we supply the device of Example 4.5 with the Büchi acceptance condition $F_0 = \{a_1\}$. That is, the resulting automaton $\mathbb{A}_0$ accepts a stream α iff the run of $\mathbb{A}_0$ passes through the state a_1 infinitely often. For instance, $\mathbb{A}_0$ will accept the word $\alpha = brgbrgbrgbrgbrgbrgb\cdots$, because the run of $\mathbb{A}_0$ is the stream $a_0a_1a_2a_2a_1a_2a_2a_1a_2a_2\ldots$ which indeed contains a_1 infinitely many times. On the other hand, as we saw already, the run of $\mathbb{A}_0$ on the stream $\alpha' = brbgbrgrgrgrgrgr\cdots$ loops in state a_2, and so α' will not be accepted.

In general, it is not hard to prove that $\mathbb{A}_0$ accepts a C-stream γ iff γ contains infinitely many b's. ◁

Example 4.13 Consider the automaton $\mathbb{A}_1$ given by the following diagram and initial state:



As an example of a Muller acceptance condition, consider the set

$$\{ \{a_0\} , \{a_g\} , \{a_b, a_g\} , \{a_b, a_r, a_g\} \}$$

The resulting automaton accepts those infinite streams in which every b is followed by a finite number of r's, followed by a g. To see this, here is a brief description of the intuitive meaning of the states:

a_0 represents the situation where the automaton has not encountered any b's;

a_f is the 'faulty' state;

a_b is the state where the automaton has just processed a b; it now has to pass through a finite sequence of r's, eventually followed by a g;

a_r represents the situation where the automaton, after seeing a b, has processed a finite, non-empty, sequence of r's;

a_g is the state where the automaton, after passing the last b, has fulfilled its obligation to process a g.

We leave the details of the proof as an exercise to the reader. $\lhd$

Example 4.14 For an example of a parity automaton, consider the transition diagram of Example 4.5, and suppose that we endow the set $\{a_0, a_1, a_2\}$ with the priority map Ω given by $\Omega(a_i) = i$. Given the shape of the transition diagram, it then follows more or less directly from the definitions that the resulting automaton accepts an infinite word over $C = \{b, r, g\}$ iff it either stays in a_0, or visits a_2 infinitely often. From this one may derive that $L_\omega(\mathbb{A})$ consists of those C-streams containing infinitely many r's or infinitely many g's (or both). $\lhd$

It is important to understand the relative strength of Muller, Büchi and parity automata when it comes to recognizing ω-languages. The Muller acceptance condition is the more fundamental one in the sense that the other two are easily represented by it.

Proposition 4.15 *There is an effective procedure transforming a deterministic Büchi stream automaton into an equivalent deterministic Muller stream automaton.*

Proof. Given a Büchi condition F on a set A, define the corresponding Muller condition $\mathcal{M}_F \subseteq \wp(A)$ as follows:

$$\mathcal{M}_F := \{B \subseteq A \mid B \cap F \neq \varnothing\}.$$

Clearly then, $Acc_{\mathcal{M}_F} = Acc_F$. It is now immediate that any Büchi automaton $\mathbb{A} = \langle A, \delta, F, a_I\rangle$ is equivalent to the Muller automaton $\langle A, \delta, \mathcal{M}_F, a_I\rangle$. QED

Proposition 4.16 *There is an effective procedure transforming a deterministic parity stream automaton into an equivalent deterministic Muller stream automaton.*

Proof. Analogous to the proof of the previous proposition, we put

$$\mathcal{M}_\Omega := \{B \subseteq A \mid \max(\Omega[B]) \text{ is even }\},$$

and leave it for the reader to verify that this is the key observation in turning a parity acceptance condition into a Muller one. QED

Interestingly enough, Muller automata can be simulated by devices with a parity condition.

Proposition 4.17 *There is an effective procedure transforming a deterministic Muller stream automaton into an equivalent deterministic parity stream automaton.*

Proof. Given a Muller automaton $\mathbb{A} = \langle A, \delta, \mathcal{M}, a_I\rangle$, define the corresponding parity automaton $\mathbb{A}' = \langle A', \delta', \Omega, a'_I\rangle$ as follows. The crucial concept used in this construction is that of *latest appearance records*. The following notation will be convenient: given a finite sequence in A^*, say, $\alpha = a_1 \dots a_n$, we let $\widetilde{\alpha}$ denote the set $\{a_1, \dots, a_n\}$, and $\alpha[\triangledown/a]$ the sequence α with every occurrence of a being replaced with the symbol $\triangledown$.

To start with, the set A' of states is defined as the collection of those finite sequences over the set $A \cup \{\triangledown\}$ in which every symbol occurs exactly once:

$$A' = \{a_1 \dots a_k \triangledown a_{k+1} \dots a_m \mid m = |A| \text{ and } A = \{a_1, \dots, a_m\}\}.$$

The intuition behind this definition is that a state in $\mathbb{A}'$ encodes information about the states of $\mathbb{A}$ that have been visited during the initial part of its run on some word. More specifically, the state $a_1 \dots a_k \triangledown a_{k+1} \dots a_m$ encodes that the states visited by $\mathbb{A}$ are $a_{n+1}, \dots, a_m$ (for some $n \leq m$, not necessarily $n = k$), and that of these, a_m is the state visited most recently, a_{m-1} the one before that, etc. The symbol $\triangledown$ marks the *previous* position of a_m in the list.

For a proper understanding of $\mathbb{A}'$ we need to go into more detail. First, for the initial position of $\mathbb{A}'$, fix some enumeration $d_1, \dots, d_m$ of A with $a_I = d_m$, and define

$$a'_I := d_1 \dots d_m \triangledown.$$

For the transition function, consider a state $\alpha = a_1 \dots a_k \triangledown a_{k+1} \dots a_m$ in A', and a color $c \in C$. To obtain the state $\delta'(\alpha, c)$, replace the occurrence of $\delta(a_m, c)$ in $a_1 \dots a_m$ with $\triangledown$, and make the state $\delta(a_m, c)$ itself the rightmost element of the resulting sequence. Thus the ∇ in the new sequence marks the latest appearance of the state $\delta(a_m, c)$. Formally, we put

$$\delta'(a_1 \dots a_k \nabla a_{k+1} \dots a_m, c) := (a_1 \dots a_m)[\nabla/\delta(a_m, c)] \cdot \delta(a_m, c).$$

(Here we include the cases where $k = 0$ or $k = m$; these cover the situations where ∇ appears at, respectively, the beginning or the end of the word.) For an example, see 4.18 below.

Now consider the runs ρ and ρ' of $\mathbb{A}$ and $\mathbb{A}'$, respectively, on some C-stream γ. Recall that $\mathit{Inf}(\rho)$ denotes the set of states of $\mathbb{A}$ that are visited infinitely often during ρ. From a certain moment on, ρ will *only* pass through states in $\mathit{Inf}(\rho)$; let $\mathbb{A}$ continue its run until it has passed through each state in $\mathit{Inf}(\rho)$ at least one more time. It is not too hard to see that there is some l such that from that same moment t on, ρ' will only pass through states of the form $a_1 \dots a_k \nabla a_{k+1} \dots a_m$ such that the states in $\mathit{Inf}(\rho)$ are those that form the final segment $a_{l+1} \dots a_m$ of the sequence $a_1 \dots a_m$.

We now arrive at the role of the special symbol ∇. Since ∇ marks the previous position of a_m, all states occurring to its right after time t must belong to the set $\mathit{Inf}(\rho)$. In other words, we have

$$\mathit{Inf}(\rho') \subseteq \{\alpha\nabla\beta \in A' \mid \widetilde{\beta} \subseteq \mathit{Inf}(\rho)\}.$$

Furthermore, among the states $\alpha\nabla\beta \in \mathit{Inf}(\rho')$, the ones with the *longest tail* β (i.e., with maximal $|\beta|$), are exactly the ones where $\mathit{Inf}(\rho)$ is *identical* to $\widetilde{\beta}$. Obviousy, these will be of interest for the definition of the acceptance condition of $\mathbb{A}'$. To make the discussion somewhat more precise, define, for a subset Q of the state space A', $\overline{Q} := \{\alpha\nabla\beta \in Q \mid |\widetilde{\beta}'| \leq |\widetilde{\beta}| \text{ for all } \alpha'\nabla\beta' \in Q\}$. That is, $\overline{Q}$ consists of the sequences $\alpha\nabla\beta \in Q$ where β takes maximal length. Then one may show that

$$\alpha\nabla\beta \in \overline{\mathit{Inf}(\rho')} \text{ implies } \widetilde{\beta} = \mathit{Inf}(\rho). \tag{40}$$

This shows how to encode the success of runs of $\mathbb{A}$ in a parity condition for $\mathbb{A}'$. Putting

$$\Omega(\alpha\nabla\beta) := \begin{cases} 2 \cdot |\beta| + 1 & \text{if } \widetilde{\beta} \notin \mathcal{M}, \\ 2 \cdot |\beta| + 2 & \text{if } \widetilde{\beta} \in \mathcal{M}, \end{cases}$$

we ensure that the states in $\overline{\mathit{Inf}(\rho')}$ receive maximal priority, and that this priority is even.

We now have the following chain of equivalences:

$$\begin{aligned} & \mathbb{A} \text{ accepts } \gamma \\ \iff & \mathit{Inf}(\rho) \in \mathcal{M} && \text{(definition acceptance } \mathbb{A}) \\ \iff & \widetilde{\beta} \in \mathcal{M} \text{ whenever } \alpha\nabla\beta \in \overline{\mathit{Inf}(\rho')} && \text{(statement (40))} \\ \iff & \max\{\Omega(\alpha\nabla\beta) \mid \alpha\nabla\beta \in \mathit{Inf}(\rho')\} \text{ is even} && \text{(as discussed above)} \\ \iff & \mathbb{A}' \text{ accepts } \gamma. && \text{(definition acceptance } \mathbb{A}') \end{aligned}$$

Clearly this establishes the equivalence of $\mathbb{A}$ and $\mathbb{A}'$. QED

Example 4.18 With $\mathbb{A}_1$ the Muller automaton of Example 4.13, here are some examples of the transition function δ' of its parity equivalent $\mathbb{A}'$:

$$\begin{array}{rclrcl}
\delta'(a_b a_r a_g a_f a_0 \nabla, b) & := & \nabla a_r a_g a_f a_0 a_b & \delta'(\nabla a_r a_g a_f a_0 a_b, b) & := & a_r a_g \nabla a_0 a_b a_f \\
\delta'(a_b a_r a_g a_f a_0 \nabla, r) & := & a_b a_r a_g a_f \nabla a_0 & \delta'(\nabla a_r a_g a_f a_0 a_b, r) & := & \nabla a_g a_f a_0 a_b a_r \\
\delta'(a_b a_r a_g a_f a_0 \nabla, g) & := & a_b a_r a_g a_f \nabla a_0 & \delta'(\nabla a_r a_g a_f a_0 a_b, g) & := & a_r \nabla a_f a_0 a_b a_g
\end{array}$$

Likewise, a few examples of the priority map:

$$\begin{array}{rcl}
\Omega(a_b a_r a_g a_f \nabla a_0) & := & 4 \\
\Omega(a_g a_f a_0 a_b \nabla a_r) & := & 3 \\
\Omega(a_f a_r a_0 \nabla a_b a_g) & := & 6 \\
\Omega(a_f a_0 \nabla a_b a_r a_g) & := & 8
\end{array}$$

As the initial state of $\mathbb{A}'$, one could for instance take the sequence $a_r a_r a_g a_f a_0 \nabla$. $\triangleleft$

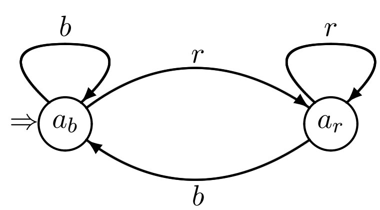
The following example shows that, in the case of deterministic stream automata, the recognizing power of Muller and parity automata is *strictly* stronger than that of Büchi automata.

Example 4.19 Consider the following language over the alphabet $C = \{b, r\}$:

$$L = \{\alpha \in C^\omega \mid r \notin \mathit{Inf}(\alpha)\}.$$

That is, L consists of those C-streams that contain at most finitely many red items (that is, the symbol r occurs at most finitely often). We will give both a Muller and a parity automaton to recognize this language, and then show that there is no Büchi automaton for L.

It is not difficult to see that there is a deterministic Muller automaton recognizing this language. Consider the automaton $\mathbb{A}_2$ given by the following diagram,



and Muller acceptance condition $\mathcal{M}_2 := \{\{a_b\}\}$. It is straightforward to verify that the run of $\mathbb{A}_2$ on an $\{b, r\}$-stream α keeps circling in a_b iff from a certain moment on, α only produces b's.

For a parity automaton recognizing L, endow the diagram above with the priority map Ω_2 given by $\Omega_2(a_b) = 0$, $\Omega_2(a_r) = 1$. With this definition, there can only be one set of states of which the maximum priority is even, namely, the singleton $\{a_b\}$. Hence, this parity acceptance condition is the same as the Muller condition $\{\{a_b\}\}$.

However, there is *no* deterministic *Büchi* automaton recognizing L. Suppose for contradiction that $L = L_\omega(\mathbb{A})$, where $\mathbb{A} = \langle A, \delta, F, a_I \rangle$ is some Büchi automaton. Since the stream

$\alpha_0 = bbb\ldots$ belongs to L, it is accepted by $\mathbb{A}$. Hence in particular, the run ρ_0 of $\mathbb{A}$ on α_0 will pass some state $f_0 \in F$ after a finite number, say n_0, of steps.

Now consider the stream $\alpha_1 = b^{n_0}rbbb\ldots$. Since runs are uniquely determined, the initial n_0 steps of the run ρ_1 of $\mathbb{A}$ on α_1 are identical to the first n_0 steps of $\mathbb{A}$ on α_0, and so ρ_1 also passes through f_0 after n_0 steps. But since α_1 belongs to L too, it too is accepted by $\mathbb{A}$. Thus on input α_1, $\mathbb{A}$ will visit a state in F infinitely often. That is, we may certainly choose an $n_1 \in \omega$ such that ρ_1 passes through some state $f_1 \in F$ after $n_0 + n_1 + 1$ steps. Now consider the stream $\alpha_2 = b^{n_0}rb^{n_1}rbbb\ldots$, and analyze the run ρ_2 of $\mathbb{A}$ on α_2. Continuing like this, we can find positive numbers $n_0, n_1, \ldots$ such that for every $k \in \omega$, the stream

$$\alpha_k = b^{n_0}rb^{n_1}\ldots rb^{n_k}rbbb\ldots \in L, \text{ for all } k. \tag{41}$$

Consider the stream

$$\alpha = (b^{n_0}r)(b^{n_1}r)\ldots(b^{n_k}r)\ldots$$

Containing infinitely many r's, α does not belong to L. Nevertheless, it follows from (41) that the run ρ of $\mathbb{A}$ on α passes through the states $f_0, f_1, \ldots$ as described above. Since F is finite, there is then at least one $f \in F$ appearing infinitely often in this sequence. Thus we have found an $f \in F$ that is passed infinitely often by ρ, showing that $\mathbb{A}$ accepts α. This gives the desired contradiction. ◁

Remark 4.20 Since it is easy to see that the complement

$$\overline{L} = \{\alpha \in C^\omega \mid r \in \mathit{Inf}(\alpha)\}$$

of the language studied in Example 4.19 *is* recognized by a Büchi automaton, the example also shows that the class of Büchi recognizable stream languages is not closed under taking complementations. ◁

4.3 Nondeterministic automata

Nondeterministic automata generalize deterministic ones in that, given a state and a color, the next state is not *uniquely* determined, and in fact need not exist at all.

Definition 4.21 Given an *alphabet* C, a *nondeterministic C-automaton* is a quadruple $\mathbb{A} = \langle A, \Delta, Acc, a_I\rangle$, where A is a finite set, $a_I \in A$ is the *initial state* of $\mathbb{A}$, $\Delta : A \times C \to \wp(A)$ its *transition function* of $\mathbb{A}$, and $Acc \subseteq A^\omega$ its *acceptance condition*. ◁

As a consequence, the run of a nondeterministic automaton on a stream is no longer uniquely determined either.

Definition 4.22 Given a nondeterministic automaton $\mathbb{A} = \langle A, \Delta, Acc, a_I\rangle$, we define the relations $\to \subseteq A \times C \times A$ and $\twoheadrightarrow \subseteq A \times C^* \times A$ in the obvious way: $a \xrightarrow{c} a'$ if $a' \in \Delta(a, c)$, $a \overset{\varepsilon}{\twoheadrightarrow} a'$ if $a = a'$, and $a \overset{wc}{\twoheadrightarrow} a'$ if there is a a'' such that $a \overset{w}{\twoheadrightarrow} a'' \xrightarrow{c} a'$. A *run* of a nondeterministic automaton $\mathbb{A} = \langle A, \Delta, Acc, a_I\rangle$ on an C-stream $\gamma = c_0c_1c_2\ldots$ is an infinite A-sequence

$$\rho = a_0a_1a_2\ldots$$

such that $a_0 = a_I$ and $a_i \xrightarrow{c_i} a_{i+1}$ for every $i \in \omega$. ◁

Now that runs are no longer unique, an automaton may have both successful and unsuccessful runs on a given stream. Consequently, there is a choice to make concerning the definition of the notion of acceptance.

Definition 4.23 A nondeterministic C-automaton $\mathbb{A} = \langle A, \Delta, Acc, a_I \rangle$ *accepts* a C-stream γ if there is a successful run of $\mathbb{A}$ on γ. $\lhd$

Further concepts, such as the language recognized by an automaton, the notion of equivalence of two automata, and the Büchi, Muller and parity acceptance conditions, are defined as for deterministic automata. Also, the transformations given in the Propositions 4.15, 4.16 and 4.17 are equivalence-preserving for nondeterministic automata just as for deterministic one. *Different* from the deterministic case, however, is that *nondeterministic* Büchi automata have the *same* accepting power as their Muller and parity variants.

Proposition 4.24 *There is an effective procedure transforming a nondeterministic Muller stream automaton into an equivalent nondeterministic Büchi stream automaton.*

Proof. Let $\mathbb{A} = \langle A, \Delta, \mathcal{M}, a_I \rangle$ be a nondeterministic Muller automaton. The idea underlying the definition of the Büchi equivalent $\mathbb{A}'$ is that $\mathbb{A}'$, while copying the behavior of $\mathbb{A}$, *guesses* the set $M = \mathit{Inf}(\rho)$ of a successful run of $\mathbb{A}$, and at a certain (nondeterministically chosen) moment confirms this choice by moving to a position of the form $(a, M, \varnothing)$. In order to make sure that not too many streams are accepted, the device has to keep track which of the states in M have been visited by $\mathbb{A}$, resetting this counter to the empty set every time when *all* M-states have been passed.

In some more detail, $\mathbb{A}'$ consists of a copy of $\mathbb{A}$, together with, for every set $M \in \mathcal{M}$, a part $\mathbb{A}_M$ which, roughly spoken, corresponds to a copy of $\mathbb{A}$ from which all states outside of M have been removed, and whose states record the part of M that recently has been visited.

$$
\begin{array}{rcl}
A' & := & A \cup \displaystyle\bigcup_{M \in \mathcal{M}} \{(a, M, P) \mid a \in M, P \subseteq M\}, \\
a'_I & := & a_I \\
\Delta'(a, c) & := & \Delta(a, c) \cup \displaystyle\bigcup_{M \in \mathcal{M}} \{(b, M, \varnothing) \mid b \in \Delta(a, c) \cap M\} \\
\Delta'((a, M, P), c) & := & \left\{ \begin{array}{ll} \{(b, M, P \cup \{a\}) \mid b \in \Delta(a, c) \cap M\} & \text{if } P \cup \{a\} \neq M, \\ \{(b, M, \varnothing) \mid b \in \Delta(a, c) \cap M\} & \text{if } P \cup \{a\} = M. \end{array} \right. \\
F & := & \displaystyle\bigcup_{M \in \mathcal{M}} \{(a, M, P) \in A' \mid P = \varnothing\}.
\end{array}
$$

We leave it as an exercise for the reader to verify that the resulting automaton is indeed equivalent to $\mathbb{A}$. QED

We now turn to the *determinization* problem for stream automata. In the case of automata operating on finite words, it is not difficult to prove that nondeterminism does not really add recognizing power: any nondeterministic automaton $\mathbb{A}$ may be 'determinized', that is, transformed into an equivalent deterministic automaton $\mathbb{A}^d$.

Remark 4.25 Finite word automata (see Example 4.9) can be determinized by a fairly simple *subset construction.*

Let $\mathbb{A} = \langle A, \Delta, F, a_I \rangle$ be a nondeterministic word automaton. A run of $\mathbb{A}$ on a finite word $w = c_1 \cdots c_n$ is defined as a finite sequence $a_0 a_1 \cdots a_n$ such that $a_0 = a_I$ and $a_i \xrightarrow{c_i} a_{i+1}$ for all $i < n$. $\mathbb{A}$ *accepts* a finite word w if there is a successful run, that is, a run $a_0 a_1 \cdots a_n$ ending in an accepting state a_n.

Given such a nondeterministic automaton, define a deterministic automaton $\mathbb{A}^+$ as follows. For the states of $\mathbb{A}^+$ we take the *macro-states* of $\mathbb{A}$, that is, the nonempty subsets of A. The deterministic transition function δ is given by

$$\delta(P, c) := \bigcup_{a \in P} \Delta(a, c).$$

In words, $\delta(P, c)$ consists of those states that can be reached from some state in P by making one a-step in $\mathbb{A}$. The accepting states of $\mathbb{A}^+$ are those macro-states that contain an accepting state from $\mathbb{A}$: $F^+ := \{P \in A^+ \mid P \cap F \neq \varnothing\}$, and its initial state is the singleton $\{a_I\}$.

In order to establish the equivalence of $\mathbb{A}$ and $\mathbb{A}^+$, we need to prove that for every word w, $\mathbb{A}$ has an accepting run on w iff the unique run of $\mathbb{A}^+$ on w is successful. The key claim in this proof is the following statement:

$$\widehat{\delta}(\{a_I\}, w) = \{a \in A \mid a_I \overset{w}{\twoheadrightarrow}_{\mathbb{A}} a\}. \tag{42}$$

stating that $\widehat{\delta}(\{a_I\}, w)$ consists of all the states that $\mathbb{A}$ can reach from a_I on input w. We leave the straightforward inductive proof of (42) as an exercise for the reader.

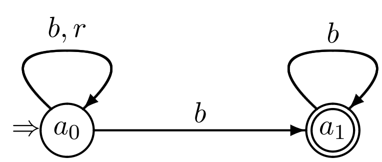
The equivalence of $\mathbb{A}$ and $\mathbb{A}^+$ then follows by the following chain of equivalences, for any finite word w: $\mathbb{A}^+$ accepts w iff $\widehat{\delta}(\{a_I\}, w) \in F^+$ iff $\widehat{\delta}(\{a_I\}, w) \cap F \neq \varnothing$ iff $a_I \overset{w}{\twoheadrightarrow}_{\mathbb{A}} a$ for some $a \in F$ iff $\mathbb{A}$ accepts w. $\triangleleft$

Unfortunately, the class of Büchi automata does not admit such a determinization procedure. As a consequence of Proposition 4.24 above, and witnessed by the Examples 4.19 and 4.26, the recognizing power of nondeterministic Büchi automata is strictly greater than that of their deterministic variants.

Example 4.26 For a nondeterministic Büchi automaton recognizing the language

$$L = \{\alpha \in C^\omega \mid r \notin \mathit{Inf}(\alpha)\}$$

of Example 4.19, consider the automaton given by the following picture:



In general, the Büchi acceptance condition $F \subseteq A$ of an automaton $\mathbb{A}$ is depicted by the set of states with *double circles*. So in this case, $F = \{a_1\}$. $\triangleleft$

There is positive news as well. A key result in automata theory states that when we turn to Muller and parity automata, nondeterminism does *not* increase recognizing power. This result follows from Proposition 4.24 and Theorem 4.27 below.

Theorem 4.27 *There is an effective procedure transforming a nondeterministic Büchi stream automaton into an equivalent deterministic Muller stream automaton.*

The *proof* of Theorem 4.27 will be given in the next section. As an important corollary we mention the following *Complementation Lemma*.

Proposition 4.28 *Let $\mathbb{A}$ be a nondeterministic Muller or parity automaton. Then there is an automaton $\overline{\mathbb{A}}$ of the same kind, such that $L_\omega(\overline{\mathbb{A}})$ is the complement of the language $L_\omega\mathbb{A}$.*

Leaving the proof of this proposition as an exercise for the reader, we finish this section with a summary of the relative power of the automata concept in the diagram below. Arrows indicate the reducibility of one concept to another, 'D' and 'ND' are short for 'deterministic' and 'nondeterministic', respectively.

$$
\boxed{\begin{array}{ccccc}
\text{D Büchi} & \Longrightarrow & \text{D Muller} & \Longleftrightarrow & \text{D parity} \\
\Downarrow & & \Updownarrow & & \Updownarrow \\
\text{ND Büchi} & \Longleftrightarrow & \text{ND Muller} & \Longleftrightarrow & \text{ND parity}
\end{array}}
$$

Having established these equivalences we naturally arrive at the following definition.

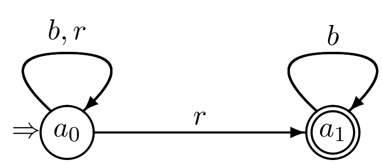
Definition 4.29 Let C be a finite set. A C-stream language $L \subseteq C^\omega$ is called *ω-regular* if there exists a C-stream automaton $\mathbb{A} = (A, \Delta, \Omega, a_I)$ such that $L = L_\omega(\mathbb{A})$, where $\mathbb{A}$ is either a (deterministic/nondeterministic) Muller or parity automaton, or a nondeterministic Büchi automaton. $\lhd$

4.4 Determinization of stream automata

This section is devoted to the proof of Theorem 4.27, which is based on a modification of the subset construction of Remark 4.25.

▶ `more information on determinization/simulation to be supplied`

Remark 4.30 This modification will have to be fairly substantial: As we will see now, Theorem 4.27 cannot be proved by a straightforward adaptation of the subset construction discussed in Remark 4.25. Consider the Büchi automaton $\mathbb{A}$ given by the following picture:



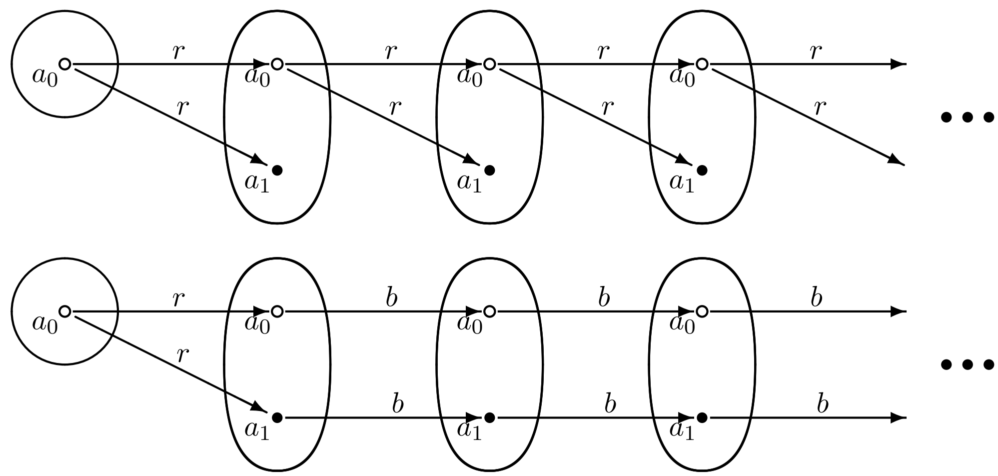
We leave it for the reader to verify that $L_\omega(\mathbb{A})$ consists of those streams of bs and rs that contain at least one and at most finitely many red items. In particular, the stream $r^\omega = rrrrr\ldots$ is rejected, while the stream $rb^\omega = rbbbb\ldots$ is accepted.

Now consider a deterministic automaton $\mathbb{A}^+$ of which the transition diagram is given by the subset construction. Then the run of the automaton $\mathbb{A}^+$ on r^ω is *identical* to its run on rb^ω:

$$a_0\{a_0, a_1\}\{a_0, a_1\}\{a_0, a_1\}\ldots$$

In other words, no matter which acceptance condition we give to $\mathbb{A}^+$, the automaton will accept either both r^ω and rb^ω, or neither. In either case $L_\omega(\mathbb{A}^+)$ will be different from $L_\omega(\mathbb{A})$.

As a matter of fact, it will be instructive to see in a bit more detail how the runs of $\mathbb{A}$ on r^ω and rb^ω, respectively, appear as 'traces' in the run of $\mathbb{A}^+$ on these two streams:



Clearly, where the second run contains one single trace that corresponds to a successful run of the automaton $\mathbb{A}$, in the first run, all traces that reach a successful state are aborted immediately. These two pictures clarify the subtle but crucial distinctions that get lost if we try to determinize via a straightforward subset construction. $\lhd$

In Safra's modification of the subset construction, the states of the deterministic automaton are *finite, structured collections of macro-states*; more specifically, if we order these macro-states by the inclusion relation we obtain a certain tree structure. The key idea underlying this modification is that at each step of the run, those elements of a macro-state that are accepting (i.e., members of the Büchi set of the original automaton), will be given some special treatment. Ultimately this enables one to single out the runs with a sequence of macro-states containing a good trace (that is, an infinite sequence of states constituting an accepting run of the nondeterministic automaton).

For the formal definition of Safra trees, we recall that we call two distinct nodes in a tree are called *siblings* if they have the same parent, and that, where $\lhd$ denotes the parent-child relation, its transitive closure denotes the ancestor/descendant relation. That is, if $s \lhd^+ t$ we call s a *descendant* of t, and t an *ancestor* of s. Furthermore, we recall that, where s and t are distinct nodes that are not related by the ancestor/descendant relation, there is a unique pair of siblings (s', t') such that s and t are either equal to or descendants of, respectively, s' and t'; we call this pair the *ancestral* sibling pair of (s, t).

Definition 4.31 An ordered tree is a structure $\langle S, r, \lhd, <_H \rangle$ such that $\langle S, \lhd \rangle$ is a tree with root r; $\lhd$ is the 'child-of' relation, with $s \lhd t$ denoting that s is a child of t; and $<_H$ is a *sibling ordering relation*, that is, a strict partial order on S that totally orders the children of every node; if $s <_H t$ we may say that s is *older* than t. Given two distinct nodes s and t such that neither $s \lhd^* t$ nor $t \lhd^* s$, we say that s is *to the left of* t if the unique ancestral sibling pair (s', t') of (s, t) is such that $s' <_H t'$.

A *Safra tree* over a set B is a pair (S, L) where S is a finite ordered tree, and $L : S \to \wp^+(B)$ is a labelling assigning a non-empty macrostate $L(s)$ to every node s in such a way that (i) for every node s, the set $\bigcup\{L(t) \mid t \lhd s\}$ is a *proper* subset of $L(s)$, and (ii) $L(s) \cap L(t) = \varnothing$ if s and t are siblings. $\lhd$

It is not hard to see that for any Safra tree (S, L) and for every state $b \in B$, b belongs to some label set of the tree iff it belongs to the label of the root. And, if b belongs to the label of the root, then there is a *unique* node $s \in S$, the so-called *lowest node of* b, such that $b \in L(s)$ but s has no child t with $b \in L(t)$. From these observations one easily derives that

$$|S| \leq |B|, \tag{43}$$

for every Safra tree over the set B.

We now turn to the details of the Safra construction.

Definition 4.32 Let $\mathbb{B}$ be a nondeterministic Büchi automaton $\mathbb{B} = \langle B, b_I, \Delta, F \rangle$. We will define a deterministic Muller automaton $\mathbb{B}^S = \langle B^S, a_I, \delta, \mathcal{M} \rangle$.

Assume that B has n states, and let $N := \{1, \ldots, 2n\}$; we will think of N as the set of *(potential) nodes* of a Safra tree. The carrier B^S will consist of the collection of all *colored Safra trees* over B, that is, all triples (S, L, θ) such that (S, L) is a Safra tree over B with $S \subseteq N$, and θ is a map coloring nodes of the tree either white or green. The initial state of $\mathbb{B}^S$ will be the Safra tree consisting of a single white node 1 labelled with the singleton $\{b_I\}$.

For the transition function on B^S, take an arbitrary colored Safra tree (S, L, θ). On input $c \in C$, the deterministic transition function δ on B^S transforms (S, L, θ) into a new colored, labelled Safra tree, by performing the sequence of actions below. (Note that at intermediate stages of this process, the structures may violate the conditions of Safra trees.)

1. *Make macro-move* Apply the power set construction to the individual nodes: for each node s, replace its label $L(s) \subseteq B$ with the set $L'(s) := \bigcup_{a \in L(s)} \Delta(a, c)$.

2. *Separate accepting states* For each node $s \in S$ such that $L'(s)$ contains accepting states, add a new[3] node $s' \in N \setminus S$ to S as the youngest child of s, and label s' with the set $Ł'(s') := L'(s) \cap F$. (Such an s' can be canonically chosen as the smallest $n \in N$ such that $n \notin S$).

[3]Observe that by (43) and the definition of N, there will always be sufficiently many nodes in N such that at least one element of N is left as a 'spare' node, possibly to be used at a later stage.

3. *Merge traces* For each node s, remove those members from its label that already belong to the label of a state to the left of s (3a). After that remove all nodes with empty labels (3b).

4. *Mark successful nodes* For each (remaining) node s of which the label is *identical* to the union of the labels of its children, remove all proper descendants of s, and *mark* s by coloring it green. All other nodes are colored white.

For the Muller acceptance condition $\mathcal{M}$ of $\mathbb{B}^S$, put $M \in \mathcal{M}$ if there is some $s \in \{0, \ldots, 2n\}$ such that s is present as a node of every tree in M, and s is colored green in some tree in M. $\triangleleft$

Example 4.33 ▶ `Example to be supplied`

$\triangleleft$

The following proposition states that the size of the Safra automaton is exponentially bounded.

Proposition 4.34 *Let $\mathbb{B}$ be a nondeterministic Büchi automaton with n states. Then $|\mathbb{B}^S|$ has at most $2^{\mathcal{O}(n*\log(n))}$ states.*

Proof. We will prove the Proposition by showing there are at most $(2n+1)^{7n}$ coloured Safra trees over a set B of size n. For this purpose we represent coloured Safra trees in terms of functions. Recall that $N = \{1, \ldots, 2n\}$ denotes the set of (potential) nodes of a Safra tree.

- To start with, the parent relation $\triangleleft$ of a Safra tree can be represented by a *parent* function $p : N \to N \cup \{0\}$ which maps every non-root node in the tree to its unique parent, and every other element of N to 0. There are at most $(2n+1)^{2n}$ of such maps.

- Similarly, the sibling order $<_H$ can be represented by a map from $N \to N \cup \{0\}$ which maps any node which has older siblings to the youngest of these, and every other node to 0. Again, there at most $(2n+1)^{2n}$ of such maps.

- The macro-state labelling L of a Safra can be represented by the function $m : B \to N \cup \{0\}$ which maps a state $b \in B$ to 0 if $b \notin L(r)$ (i.e., b is not present in the Safra tree), and to the unique-lowest node s in the tree such that $b \in L(s)$, otherwise. The number of these maps is therefore bounded by $(2n+1)^n$.

- Finally, for reasons of similarity, the colouring map θ can be represented as a map from N to $N \cup \{0\}$ which maps $s \in N$ to 0 if it coloured green, and to 1 if it is either white or not present in the tree. Hence there are at most $(2n+1)^{2n}$ of such maps.

Every coloured Safra tree can thus be represented as a quadruple of maps from either N or B to $N \cup \{0\}$, and so the number of these trees is bounded by $(2n+1)^{2n} * (2n+1)^{2n} * (2n+1)^n * (2n+1)^{2n} = (2n+1)^{7n}$. QED

It is obvious from the construction that $\mathbb{B}^S$ is a deterministic automaton, so what is left of the proof of Theorem 4.27 is to establish the equivalence of $\mathbb{B}$ and $\mathbb{B}^S$.

Proposition 4.35 *Let $\mathbb{B}$ be a nondeterministic Büchi automaton. Then*

$$L_\omega(\mathbb{B}) = L_\omega(\mathbb{B}^S).$$

Proof.(Sketch) For the inclusion $\subseteq$, assume that there is a successful run $\rho = b_0 b_1 \ldots$ of $\mathbb{B}$ on some C-stream $\gamma = c_0 c_1 \ldots$. Consider the (unique) run $\sigma = (S_0, L_0, \theta_0)(S_1, L_1, \theta_1) \ldots$ of $\mathbb{B}^S$ on γ. Here each (S_i, L_i, θ_i) is a Safra tree with labeling L_i and coloring θ_i. We claim that there is an object s which after some initial phase belongs to each Safra tree of σ, and which is marked green infinitely often. The basic idea underlying the proof of this claim is to 'follow' the run ρ as a trace through the successive trees of σ.

First note that at every stage i, the state b_i of ρ belongs to the label $L_i(r_i)$ of the root r_i of the Safra tree S_i. It follows that the root always has a non-empty label, and hence it is never removed; thus we have $r_0 = r_1 = \ldots$, and so, with $r := r_0$, we have already found a node r such that r is present in every Safra tree in $\mathit{Inf}(\sigma)$. Now if r is colored green infinitely often, we are done.

So suppose that this is not the case. In other words, after a certain moment i, r will no longer be marked. Since $\rho = (b_i)_{i\in\omega}$ is by assumption a successful run of $\mathbb{B}$, it passes infinitely often through a successful state. Hence we may consider the first time $j > i$ for which b_j is an accepting state. But if $b_j \in F$, then in step 2 of stage j it has been put in the label set of a new child, say, s, of r. In step 3a, b_j may be removed from the label set of s, but only in case it was already present in the label set of an older sibling of s. It is not hard to see that in step 3b or 4, b_j will not be removed from the label sets it belongs to after step 3a.

We claim that in fact

$$\text{for all } k \geq j,\ b_k \in L_k(s_k),\ \text{for some child } s_k \text{ of } r. \tag{44}$$

The proof of this claim rests on the observation that b_k can only fail to be a member of the set $\bigcup\{L_k(s) \mid s \lhd_k r\}$ in case r is a successful node in S_k, and we assumed that this was not the case. (Here $\lhd_k$ denotes the child relation in the Safra tree S_k.) Now note that the merging of traces (as described in step 3a of the procedure) may cause states to be moved to the label set of a sibling, but only to an older one. Such a shift can thus only happen finitely often, so that after some stage j_1 there is a node s such that

$$\text{for all } k > j_1 : \quad s \in S_k, s \lhd_k r,\ \text{and } b_k \in L_k(s). \tag{45}$$

We can now repeat the argument with this s taking the role of r: either s itself is marked green infinitely often, or eventually, at some stage l, the ρ-state $b_l \in F$ will be placed at the next level, and remain there. Since the depth of the Safra trees involved is bounded, there must be some node s which after some initial phase belongs to each Safra tree in σ, and which is marked infinitely often.

For the opposite inclusion $\supseteq$, suppose that the (unique) run $\sigma = (S_0, L_0, \theta_0)(S_1, L_1, \theta_1) \ldots$ of $\mathbb{B}^S$ on the input stream $\gamma = c_0 c_1 \ldots$ is successful. Then by definition there is some node $s \in N = \{0, \ldots, 2n\}$ which after some initial phase will belong to each Safra tree in σ and which will subsequently be marked green infinitely often, say at the stages $k_1 < k_2 < \cdots$. For each $i > 0$, let A_i denote the macro-state of s at stage k_i, that is: $A_i := L_{k_i}(s)$.

For natural numbers p and q, let $\gamma[p,q)$ denote the finite word $c_p \cdots c_{q-1}$, so that γ is equal to the infinite concatenation

$$\gamma = \gamma[0,k_1) \cdot \gamma[k_1,k_2) \cdot \gamma[k_2,k_3) \cdots$$

Since our construction is a refinement of the standard subset construction of Remark 4.25, by (42) it easily follows from the definitions of δ that for every state $a \in A_1$ there is a $\gamma[0,k_1)$-labeled path from b_I to a, or briefly:

$$\text{for all } a \in A_1 \text{ we have } b_I \overset{\gamma[0,k_1)}{\twoheadrightarrow} a. \tag{46}$$

With a little more effort, crucially involving the conditions for marking nodes, and the rules governing the creation and maintenance of nodes, one may prove that

$$\text{for all } i > 0 \text{ and for all } a \in A_{i+1} \text{ there is an } a' \in A_i \text{ such that } a' \overset{\gamma[k_i,k_{i+1})}{\twoheadrightarrow_F} a. \tag{47}$$

Here $a' \overset{\gamma[k_i,k_{i+1})}{\twoheadrightarrow_F} a$ means that there is a $\gamma[k_i,k_{i+1})$-labelled path from a' to a which passes through some state in F. Details of this proof are left as an exercise to the reader.

The remainder of the proof consists of finding a successful run of $\mathbb{B}$ on γ as the concatenation of a run segment given by (46) and infinitely many run segments given by (47). For this we use König's Lemma.

Defining $A_0 := \{b_I\}$, we will construct a tree, all of whose nodes are pairs of the form (a,i) with $a \in A_i$. As the (unique) parent of a node $(a,i+1)$ we pick one of the pairs (a',i) given by (46) (in case $i = 0$) or (47) (in case $i > 0$). Obviously this is a well-formed, infinite, finitely branching tree. So by König's Lemma, there is an infinite branch $(a_0,0)(a_1,1)\cdots$. By construction, we have $a_0 = b_I$, while for each $i \geq 0$ there is a $\gamma[k_i,k_{i+1})$-labelled path in $\mathbb{B}$ from a_i to a_{i+1} which passes through some accepting state of $\mathbb{B}$. The infinite concatenation of these paths gives a run of $\mathbb{B}$ on γ, which visits infinitely often an accepting state of $\mathbb{B}$, and hence by finiteness of B, it visits some state of $\mathbb{B}$ infinitely often. Clearly then this run is accepting. QED

4.5 Logic and automata

- ▶ `discuss the relation between stream automata, the linear` μ`-calculus, and monadic second-order logic;`
- ▶ `discuss linear time logic`

4.6 A coalgebraic perspective

In this section we introduce a coalgebraic perspective on stream automata. We have two reasons for doing so. First, we hope that this coalgebraic presentation will facilitate the introduction, further on, of automata operating on different kinds of structures. And second, we also believe that the coalgebraic perspective, in which the similarities between automata and the objects they classify comes out more clearly, makes it easier to understand some of the fundamental concepts and results in the area.

In this context, it makes sense to consider a slightly wider class than streams only.

Definition 4.36 A *C-flow* is a pair $\mathbb{S} = \langle S, \sigma \rangle$ with $\sigma : S \to C \times S$. Often we will write $\sigma(s) = (\sigma_C(s), \sigma_0(s))$. If we single out an (initial) state $s_0 \in S$ in such a structure, we obtain a *pointed C-flow* $(\mathbb{S}, s_0)$. $\lhd$

Example 4.37 Streams over an alphabet C can be seen as pointed C-flows: simply identify the word $\gamma = c_0 c_1 c_2 \ldots$ with the pair $(\langle \omega, \lambda n.(c_n, n+1) \rangle, 0)$. Conversely, with any pointed flow $\langle \mathbb{S}, s \rangle$ we may associate a unique stream $\gamma_{\mathbb{S},s}$ by inductively defining $s_0 := s$, $s_{i+1} := \sigma_0(s_i)$, and putting $\gamma_{\mathbb{S}}(n) := \sigma_C(s_n)$. $\lhd$

It will be instructive to define the following notion of equivalence between flows. As its name already indicates, we are dealing with the analog of the notion of a bisimulation between two Kripke models. Since flows, having a deterministic transition structure, are less complex objects than Kripke models, the notion of bisimulation is also, and correspondingly, simpler.

Definition 4.38 Let $\mathbb{S}$ and $\mathbb{S}'$ be two C-flows. Then a nonempty relation $Z \subseteq S \times S'$ is a *bisimulation* if the following holds, for every $(s, s') \in Z$:

(color) $\sigma_C(s) = \sigma'_C(s')$;

(successor) $(\sigma_0(s), \sigma'_0(s')) \in Z$.

Two pointed flows $(\mathbb{S}, s)$ and $(\mathbb{S}', s')$ are called *bisimilar*, notation: $\mathbb{S}, s \leftrightarroweq \mathbb{S}', s'$ if there is some bisimulation Z linking s to s'. In case the flows $\mathbb{S}$ and $\mathbb{S}'$ are implicitly understood, we may drop reference to them and simply call s and s' bisimilar. $\lhd$

As an example, it is not hard to see that any pointed flow $(\mathbb{S}, s)$ is bisimilar to the stream $\gamma_{\mathbb{S},s}$ that we may associate with it (see Example 4.37). Restricted to the class of streams, bisimilarity means *identity*.

Definition 4.39 A stream is called *regular* if it is bisimilar to a finite pointed flow. $\lhd$

Associated is a new perspective on nondeterministic stream automata which makes them very much *resemble* these flows. Roughly speaking the idea is this. Think of establishing a bisimulation between two pointed flows in terms of one structure $\langle A, a_I, \alpha \rangle$ *classifying* the other, $\langle S, s_C, \sigma \rangle$.

Now on the one hand make a restriction in the sense that the classifying flow must be finite, but on the other hand, instead of demanding its transition function to be of the form $\alpha : A \to C \times A$, allow objects $\alpha(a)$ to be *sets* of pairs in $C \times A$, rather than single pairs. That is, introduce *non-determinism* by letting the transition map Δ of $\mathbb{A}$ be of the form

$$\Delta : A \to \wp(C \times A). \tag{48}$$

Remark 4.40 This presentation (48) of nondeterminism is completely *equivalent* to the one given earlier. The point is that there is a natural bijection between maps of the above kind, and the ones given in Definition 4.21 as the transition structure of nondeterministic automata:

$$A \to \wp(C \times A) \;\cong\; (A \times C) \to \wp(A). \tag{49}$$

To see why this is so, an easy proof suffices. Using the principle of currying we can show that

$$A \to ((C \times A) \to 2) \cong (A \times C \times A) \to 2 \cong (A \times C) \to (A \to 2),$$

where the first and last set can be identified with respectively the left and right hand side of (49) using the bijection between subsets and their characteristic functions.

Concretely, we may identify a map $\Delta : (A \times C) \to \wp(A)$ with the map $\Delta' : A \to \wp(C \times A)$ given by

$$\Delta'(a) := \{(c, a') \mid a' \in \Delta(a, c)\}. \tag{50}$$

$\lhd$

Thus we arrive at the following reformulation of the definition of nondeterministic automata. Note that with this definition, a stream automaton can be seen as a kind of 'multi-stream' in the sense that every state harbours a *set* of potential 'local realizations' as a flow. Apart from this, an obvious difference with flows is that stream automata also have an acceptance condition.

Definition 4.41 A *nondeterministic C-stream automaton* is a quadruple $\mathbb{A} = \langle A, \Delta, Acc, a_I\rangle$ such that $\Delta : A \to \wp(C \times A)$ is the *transition function*, $Acc \subseteq A^\omega$ is the *acceptance condition*, and $a_I \in A$ is the *initial state* of the automaton. $\lhd$

Finally, it makes sense to formulate the notion of an automaton *accepting* a flow in terms that are related to that of establishing the existence of a bisimulation. The nondeterminism can nicely be captured in game-theoretic terms — note however, that here we are dealing with a single player only.

In fact, bisimilarity between two pointed flows can itself be captured game-theoretically, using a trivialized version of the bisimilarity game for Kripke models of Definition 1.26. Consider two flows $\mathbb{A}$ and $\mathbb{S}$. Then the *bisimulation game* $\mathcal{B}(\mathbb{A}, \mathbb{S})$ between $\mathbb{A}$ and $\mathbb{S}$ is defined as a board game with positions of the form $(a, s) \in A \times S$, all belonging to $\exists$. At position (a, s), if a and s have a different color, $\exists$ loses immediately; if on the other hand $\alpha_C(a) = \sigma_C(s)$, then as the next position of the match she 'chooses' the pair consisting of the successors of a and s, respectively. These rules can concisely be formulated as in the following Table:

Position	Player	Admissible moves
$(a, s) \in A \times S$	$\exists$	$\{(\alpha_0(a), \sigma_0(s)) \mid \alpha_C(a) = \sigma_C(s)\}$

Finally, the winning conditions of the game specify that $\exists$ wins all infinite games. We leave it for the reader to verify that a pair $(a, s) \in A \times S$ is a winning position for $\exists$ iff a and s are bisimilar.

In order to proceed, however, we need to make a slight modification. We add positions of the form $(\alpha, s) \in (C \times A) \times S$, and insert an 'automatic' move immediately after a basic position, resulting in the following Table.

Position	Player	Admissible moves
$(a, s) \in A \times S$	-	$\{(\alpha(a), s)\}$
$(\alpha, s) \in (C \times A) \times S$	$\exists$	$\{(\alpha_0, \sigma_0(s)) \mid \alpha_C = \sigma_C(s)\}$

The acceptance game of a nondeterministic automaton $\mathbb{A}$ and a flow $\mathbb{S}$ can now be formulated as a natural generalization of this game.

Definition 4.42 Given a nondeterministic C-stream automaton $\mathbb{A} = \langle A, a_I, \Delta, Acc\rangle$ and a pointed flow $\mathbb{S} = \langle S, s_0, \sigma\rangle$, we now define the *acceptance game* $\mathcal{A}(\mathbb{A}, \mathbb{S})$ as the following board game.

Position	Player	Admissible moves
$(a, s) \in A \times S$	$\exists$	$\{(\alpha, s) \in (C \times A) \times S \mid \alpha \in \Delta(a)\}$
$(\alpha, s) \in (C \times A) \times S$	$\exists$	$\{(\alpha_0, \sigma_0(s)) \mid \alpha_C = \sigma_C(s)\}$

Table 7: Acceptance game for nondeterministic stream automata

Its positions and rules are given in Table 7, whereas the winning conditions of infinite matches are specified as follows. Given an infinite match of this game, first select the sequence

$$(a_0, s_0)(a_1, s_1)(a_2, s_2)\ldots$$

of *basic positions*, that is, the positions reached during play that are of the form $(a, s) \in A \times S$. Then the match is winning for $\exists$ if the 'A-projection' $a_0a_1a_2\ldots$ of this sequence belongs to Acc. $\lhd$

Definition 4.43 A nondeterministic C-stream automaton $\mathbb{A} = \langle A, a_I, \Delta, Acc\rangle$ *accepts* a pointed flow $\mathbb{S} = \langle S, s_0, \sigma\rangle$ if the pair (a_I, s_0) is a winning position for $\exists$ in the game $\mathcal{A}(\mathbb{A}, \mathbb{S})$. $\lhd$

The following proposition states that the two ways of looking at nondeterministic automata are equivalent.

Proposition 4.44 *Let $\mathbb{A} = \langle A, a_I, \Delta, Acc\rangle$, with $\Delta : (A \times C) \to \wp(A)$ be a nondeterministic C-automaton, and let $\mathbb{A}'$ be the nondeterministic C-stream automaton $\langle A, a_I, \Delta', Acc\rangle$, where $\Delta' : A \to \wp(C \times A)$ is given by (50). Then $\mathbb{A}$ and $\mathbb{A}'$ are equivalent.*

In the sequel we will *identify* the two kinds of nondeterministic automata, speaking of the *coalgebraic presentation* $\langle A, a_I, \Delta' : A \to \wp(C \times A), Acc\rangle$ of an automaton $\langle A, a_I, \Delta : (A \times C) \to \wp(A), Acc\rangle$.

Notes

The idea to use finite automata for the classification of infinite words originates with Büchi. In [4] he used stream automata with (what we now call) a Büchi acceptance condition to prove the decidability of the second-order theory of the natural numbers (with the successor relation). In the subsequent development of the theory of stream automata, other acceptance conditions were introduced. The Muller condition is named after the author of [15]. The invention of the parity condition, which can be seen as a refinement of the Rabin condition, is usually attributed to Emerson & Jutla [6], Mostowski [14], and/or Wagner.

The first construction of a deterministic equivalent to a nondeterministic Muller automaton was given by McNaughton [12]. The construction we presented in section 4.4 is due to Safra [20]. Finally, the coalgebraic perspective on stream automata presented in the final section of this chapter is the author's.

Exercises

Exercise 4.1 Provide Büchi automata recognizing exactly the following stream languages:

(a) $L_a = \{\alpha \in \{a,b,c\}^\omega \mid a \text{ and } b \text{ occur infinitely often in } \alpha\}$

(b) $L_b = \{\alpha \in \{a,b,c\}^\omega \mid \text{any } a \text{ in } \alpha \text{ is eventually followed by a } b\}$

(c) $L_c = \{\alpha \in \{a,b\}^\omega \mid \text{between any two } a\text{'s is an even number of } b\text{'s}\}$

(d) $L_d = \{\alpha \in \{a,b,c\}^\omega \mid ab \text{ and } cc \text{ occur infinitely often in } \alpha\}$

Exercise 4.2 Let C be a finite set. Show that the class of ω-regular languages over C is closed under the Boolean operations, i.e., show that

(a) If $L \subseteq C^\omega$ is ω-regular then its complement $\overline{L} := \{\gamma \in C^\omega \mid \gamma \notin L\}$ is ω-regular.

(b) If L_1 and L_2 are ω-regular C-stream languages, then $L_1 \cup L_2$ is ω-regular.

(c) If L_1 and L_2 are ω-regular C-stream languages, then $L_1 \cap L_2$ is ω-regular.

Exercise 4.3 Observe that Büchi automata can also be seen as finite automata operating on *finite* words (see Example 4.9.

(a) Show the following, for any deterministic Büchi automaton $\mathbb{A}$:

$$L_\omega(\mathbb{A}) = \{\alpha \in \Sigma^\omega \mid \text{infinitely many prefixes of } \alpha \text{ belong to } L(\mathbb{A})\}.$$

(b) Does this hold for nondeterministic Büchi automata as well?

Exercise 4.4 Let C and D be finite sets and let $f : C \to D$ be a function. The function f can be extended to a function $\overline{f} : C^\omega \to D^\omega$ in the obvious way by putting $\overline{f}(\gamma) := f(c_0)f(c_1)f(c_2)\ldots \in D^\omega$ for any C-stream $\gamma \in C^\omega$. For a given C-stream language $L \subseteq C^\omega$ we define

$$\overline{f}(L) := \{\overline{f}(\gamma) \mid \gamma \in L\} \subseteq D^\omega.$$

(a) Show that $L \subseteq C^\omega$ is ω-regular implies $f(L) \subseteq D^\omega$ is ω-regular.

(b) Show that there is a C-stream language $L \subseteq C^\omega$ such that $L = L_\omega(\mathbb{A})$ for some *deterministic* Büchi automaton $\mathbb{A}$ and such that $f(L) \subseteq D^\omega$ is not recognizable by any deterministic Büchi automaton.

Exercise 4.5 Prove that nondeterministic Büchi automata have the same recognizing power as their Muller variants by showing that the automata $\mathbb{A}'$ and $\mathbb{A}$ in the proof of Proposition 4.24 are indeed equivalent.

Exercise 4.6 Consider the language L_d of exercise 4.1.

(a) Give a clear description of the complement $\overline{L_d}$ of L_d.

(b) Give a nondeterministic Büchi automaton recognizing exactly the language $\overline{L_d}$.

(c) Prove that there is no deterministic Büchi automaton recognizing the language $\overline{L_d}$. (Hint: use the theorem from Exercise 4.3.)

Exercise 4.7 Provide deterministic Muller automata recognizing the following languages:

(a) L_d of exercise 4.1.

(b) $L_a = \{\alpha \in \{a,b,c\}^\omega \mid$ between every pair of a's is an occurrence of bb or cc $\}$.

Exercise 4.8 (regularity) Let C be a finite set, and let $L \subseteq C^\omega$ be a stream language over C. Prove that if L is ω-regular, then it contains a stream of the form uv^ω where $u \in C^*$ and $v \in C^+$.

Exercise 4.9 Describe the languages that are recognized by the following Muller automata (presented in tabular form, with $\Rightarrow$ indicating the initial state):

(a)

$\mathbb{A}$	a	b
$\Rightarrow$ q_0	q_1	q_2
q_1	q_0	q_2
q_2	q_1	q_0

with $\mathcal{F} := \{\{q_0, q_1\}, \{q_0, q_2\}\}$.

(b) The same automaton as in (a) but with $\mathcal{F} := \{\{q_1, q_2\}, \{q_0, q_1, q_2\}\}$.

(c)

$\mathbb{A}$	a	b	c
$\Rightarrow$ q_0	q_1	q_0	q_0
q_1	q_0	q_2	q_0
q_2	q_0	q_0	q_3
q_3	q_0	q_0	q_0

with $\mathcal{F} := \{\{q_0\}, \{q_0, q_1\}, \{q_0, q_1, q_2\}\}$.

Exercise 4.10 Prove (47) in the proof of Proposition 4.35. That is, show that

$$\text{for all } i > 0 \text{ and for all } a \in A_{i+1} \text{ there is an } a' \in A_i \text{ such that } a' \overset{\gamma[k_i,k_{i+1})}{\twoheadrightarrow_F} a.$$

Can you also prove that, conversely,

$$\text{for all } i > 0 \text{ and for all } a \in A_i \text{ there is an } a' \in A_{i+1} \text{ such that } a' \overset{\gamma[k_i,k_{i+1})}{\twoheadrightarrow_F} a?$$

References

[1] P. Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, chapter C.5, pages 739–782. North-Holland Publishing Co., Amsterdam, 1977.

[2] J. van Benthem. *Modal Correspondence Theory.* PhD thesis, Mathematisch Instituut & Instituut voor Grondslagenonderzoek, University of Amsterdam, 1976.

[3] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic.* Number 53 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

[4] J.R. Büchi. On a decision method in restricted second order arithmetic. In E. Nagel, editor, *Proceedings of the International Congress on Logic, Methodology and the Philosophy of Science*, pages 1–11. Stanford University Press, 1962.

[5] A. Chagrov and M. Zakharyaschev. *Modal Logic*, volume 35 of *Oxford Logic Guides.* Oxford University Press, 1997.

[6] E.A. Emerson and C.S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *Proceedings of the 32nd Symposium on the Foundations of Computer Science*, pages 368–377. IEEE Computer Society Press, 1991.

[7] D. Janin and I. Walukiewicz. Automata for the modal μ-calculus and related results. In *Proceedings of the Twentieth International Symposium on Mathematical Foundations of Computer Science, MFCS'95*, volume 969 of *LNCS*, pages 552–562. Springer, 1995.

[8] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional μ-calculus w.r.t. monadic second-order logic. In *Proceedings of the Seventh International Conference on Concurrency Theory, CONCUR '96*, volume 1119 of *LNCS*, pages 263–277, 1996.

[9] B. Knaster. Un théorème sur les fonctions des ensembles. *Annales de la Societé Polonaise de Mathematique*, 6:133–134, 1928.

[10] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[11] D. Kozen and R. Parikh. A decision procedure for the propositional μ-calculus. In *Proceedings of the Workshop on Logics of Programs 1983*, LNCS, pages 313–325, 1983.

[12] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:521–530, 1966.

[13] L. Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96:277–317, 1999. (Erratum published *Ann.P.Appl.Log.* 99:241–259, 1999).

[14] A.W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In A. Skowron, editor, *Computation Theory*, LNCS, pages 157–168. Springer-Verlag, 1984.

[15] D.E. Muller. Infinite sequences and finite machines. In *Proceedings of the 4th IEEE Symposium on Switching Circuit Theory and Logical Design*, pages 3–16, 1963.

[16] D. Niwiński. On fixed point clones. In L. Kott, editor, *Proceedings of the 13th International Colloquium on Automata, Languages and Programming (ICALP 13)*, volume 226 of *LNCS*, pages 464–473, 1986.

[17] D. Park. Concurrency and automata on infinite sequences. In *Proceedings 5th GI Conference*, pages 167–183. Springer, 1981.

[18] A. Pnueli. The temporal logic of programs. In *Proc. 18th Symp. Foundations of Computer Science*, pages 46–57, 1977.

[19] V.R. Pratt. Semantical considerations on Floyd-Hoare logic. In *Proc. 17th IEEE Symposium on Computer Science*, pages 109–121, 1976.

[20] S. Safra. On the complexity of ω-automata. In *Proceedings of the 29th Symposium on the Foundations of Computer Science*, pages 319–327. IEEE Computer Society Press, 1988.

[21] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[22] I. Walukiewicz. Completeness of Kozen's axiomatisation of the propositional μ-calculus. In *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science (LICS'95)*, pages 14–24. IEEE Computer Society Press, 1995.

[23] I. Walukiewicz. Completeness of Kozen's axiomatisation of the propositional μ-calculus. *Information and Computation*, 157:142–182, 2000.

[24] W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200:135–183, 1998.