Master's Thesis

Decidability of S1S in Constructive Type Theory

Moritz Lichter

Advisor
Prof. Dr. Gert Smolka

Reviewers
Prof. Dr. Gert Smolka
Prof. Bernd Finkbeiner, Ph.D.

submitted on
25th July, 2017

Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science

Statement in Lieu of an Oath

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, _____ _____

Date Signature

Abstract

We study monadic second order logic of $(\mathbb{N}, <)$, short S1S, in the constructive type theory of Coq. We focus on decidability of satisfiability and on logical decidability of satisfaction. We show both properties by translating S1S formulas to Büchi automata. This translation requires complementation of Büchi automata. While the complementation operation can be defined, proving it correct is not possible constructively.

We localize the property not provable in general and separate it in admissible sequence structures. AS structures come with their own representation of infinite words and a proof for the separated property. We verify the translation of formulas to Büchi automata and show decidability of satisfiability and logical decidability of satisfaction for all AS structures.

We consider two AS structures: Ultimately periodic words are admissible constructively. All infinite words are only admissible assuming Additive Ramsey. Additive Ramsey is a weakening of Ramsey's Theorem. Given Additive Ramsey, an S1S formula is ultimately periodically satisfiable if and only if it is satisfiable in general. Additive Ramsey is a necessary assumption because Additive Ramsey, closure under complement of Büchi automata, and logical decidability of satisfaction in S1S are equivalent. Moreover, the three properties are independent in constructive type theory. All results are formalized and verified in a Coq development of about 7000 lines.

Acknowledgments

I want to thank Prof. Gert Smolka for offering me the chance to work on this interesting topic. Especially, I am grateful for many interesting discussions, which helped me to view problems in a different light, constructive criticism, advice, and his patience for me and my writing.

Moreover, I want to thank Kathrin Stark for reading my thesis with utmost care, Dominik Kirst for listening to me when I was explaining things way too complicated, Yannik Forster for helping me making Coq do what I want, and my other colleagues for discussions, ideas, and the nice working environment.

Finally, I want to thank Prof. Gert Smolka and Prof. Bernd Finkbeiner for reviewing this thesis.

Contents

Chapter 1

Introduction

Monadic second order logic with one successor (short S1S) is a well-known and powerful logic. Decidability of satisfiability of S1S formulas dates back to Büchi [3] in 1962. He translated formulas to NFAs on infinite word, now known as Büchi automata. Büchi automata are useful for formal verification and much work was done on Büchi automata but on monadic second order logics, too.

We formalize S1S in the constructive type theory of Coq [20] and show decidability of satisfiability and logical satisfiability of satisfaction. For this we use a minimal system of S1S without first order variables and translate formulas to Büchi automata. For that purpose, we formalize infinite words as sequences, which are functions from $\mathbb{N}$ to some alphabet. Büchi automata are merely NFAs with Büchi acceptance for sequences.

Sequences raise several problems constructively. In contrast to (finite) strings, many properties of sequences are undecidable. Because constructive logic does not satisfy the law of excluded middle, we need to prove or disprove that a sequence fulfills a property, e.g. whether a sequence is contained in the language of a Büchi automaton. This is closely connected to the question whether satisfaction in S1S is logically decidable in a constructive setting, i.e., whether $\varphi \vee \neg\varphi$ is a tautology in S1S.

The translation of formulas to Büchi automata uses closure operations of Büchi automata. The key is complementation. Complementation of Büchi automata is still a field with ongoing work, e.g. Tsai et al. [23] compare different complementation approaches. We follow the so-called Ramsey-based approach already appearing in Büchi's seminal paper: For every Büchi automaton there are finitely many languages covering all sequences. The union of the languages disjoint from the language of the Büchi automaton yields its complement. It is possible to implement the complementation operation, but it cannot be proven correct constructively without assuming Additive Ramsey, short AR. AR is a restricted version of Ramsey's theorem. The idea of AR already appeared in Büchi [4] and the term "additive" was

used in Shelah [17]. We conjecture that AR is weaker than excluded middle and show that AR is independent in constructive type theory. The Ramsey-based complementation approach can be used to show that the word problem of Büchi automata is logically decidable.

We localize the key properties not provable constructively and separate them in admissible sequence structures, short AS structures. An AS structure provides proofs for these properties. The important property is the conclusion of a lemma called *totality* needed to prove correctness of complementation. AS structures only provide proofs, but they have no impact on the constructed Büchi automata. The translation from S1S to Büchi automata is done for an arbitrary AS structure. Then an interpretation satisfies a formula if and only if a derived sequence is accepted by the constructed Büchi automaton. Decidability of language emptiness of Büchi automata translates to decidability of satisfiability of S1S formulas. Logically decidability of the word problem of Büchi automata translates to satisfaction in S1S: satisfaction in S1S is logically decidable, so S1S admits double negation and $\varphi \vee \neg\varphi$ is a tautology.

We consider two AS structures. The ω-structure represents all sequences. We need AR to prove that the ω-structure is admissible. The second and more restricted AS structure represents only ultimately periodic sequences, short UP sequences. UP sequences are sequences of the form xy^ω and can be represented finitely by the two strings x and y. The UP-structure can be proven to be admissible completely constructively. Hence, we can view S1S constructively using UP sequences or more classically assuming AR. With AR both AS structures agree on satisfiability: a formula is satisfiable in the ω-structure if and only if it is satisfiable in the UP-structure. This corresponds to the well-known fact that any nonempty language of a Büchi automaton contains an UP sequence.

We show that AR is a necessary assumption. For this purpose, we use Ramseyan factorizations (e.g. occurring in Perrin and Pin [14]) and show that the following propositions are equivalent:

AR: Additive Ramsey: For every additive and finite coloring of pairs of numbers, there are infinitely many numbers such that all pairs of distinct numbers are colored equally.

RF: Every sequence admits a Ramseyan factorization.

BC: Büchi automata are closed under complement and the word problem for Büchi automata is logically decidable.

BU: If the languages of two Büchi automata contain the same UP sequences, then the languages are equal.

SL: Satisfaction in S1S is logically decidable.

By the prior argumentation, AR implies BC and SL. With basic facts about finite semigroups AR can be shown equivalent to RF. Equivalence of BC and BU is established using results for UP-sequences. Because we have a constructive proof that complementation is correct for UP sequences, BC and BU are equivalent. To show that BC implies RF, we give a Büchi automaton accepting sequences admitting a Ramseyan factorization. Lastly, we show that SL implies RF. The proof follows ideas given by Khoussainov and Nerode [10] and by Shelah [17]. Proving gets simpler when using S1S with first order variables. We give a (well-known) reduction from S1S with first and second order variables to S1S only with second order variables using singleton sets. We show that excluded middle implies RF and that RF implies Markovs' principle to show that RF is independent. For an overview of the implications see Figure 9.1 on page 71.

In this work we develop the mentioned results in informal mathematical language based on our formal development in the constructive type theory of Coq. This work and the formal development are aligned. Occasionally, we hide some complications of the formal development or uncommon definitions, which make proving there easier. All presented theorems and lemmas are formalized and proven in our development. Whenever appropriate, we point out differences between the formal development and this thesis. The formal development is available at `http://www.ps.uni-saarland.de/~lichter/master/`.

This work is organized as follows: We start with preliminaries of constructive type theory and formalization of sequences in Chapter 2. In Chapter 3 we introduce AR and RF and show that they are equivalent and independent in constructive type theory. In Chapter 4 we define Büchi acceptance for NFAs. We show selected properties of regular languages and prove that emptiness for Büchi automata is decidable. In Chapter 5 we prove correctness of all needed closure operations of Büchi automata excluded complementation. Complementation is mostly covered in Chapter 6. We define the complementation operation and give the important correctness proofs for sequences. The final correctness proof of the complementation construction is deferred to Chapter 7. There we introduce admissible sequence structures and give correctness proofs of all closure operation for AS structures. We show that AR implies BC and that BC is equivalent to BU. Then we define S1S in Chapter 8. We give two systems MSO and MSO_0 with and without first order variables and show that MSO can be reduced to MSO_0. We prove that the reduction from MSO_0 formulas to Büchi automata is correct and show that in MSO and MSO_0 satisfiability is decidable and that satisfaction is logically decidable. We instantiate these results for the ω-structure and the UP-structure and show that under AR both structures agree on satisfiability. In Chapter 9 we establish the equivalence between AR, RF, BC, BU, and SL by giving the missing implications from BC to RF and from SL to RF.

Chapter 2

Preliminaries

In the following we introduce preliminaries from constructive type theory, strings, sequences, and languages.

2.1 Type Theory

We formalize our results in the constructive and intensional type theory of Coq. Because the logic is constructive, it does not satisfy the law of excluded middle, and because it is intensional, pointwise equal functions or pointwise equivalent predicates are not necessarily equal. Because there is no excluded middle, decidable predicates are important. For some type X, a predicate p on X is **decidable** if it is possible to compute whether px holds or px does not hold meaning that there is a function of type $\forall x.\ (px) + (\neg(px))$.

A type X is **discrete** if equality on X is decidable. A **finite type** is a discrete type together with a list containing all elements of X. We write $|X|$ for the number of elements of X. Finite and discrete types are closed under forming product, sum, and option types: if X and Y are finite types, so are $X \times Y$, $X + Y$, and $X_\perp$.

If X is a finite type, then the Σ-Type $\Sigma x.px$ is finite if p is proof irrelevant. All decidable predicates can be turned to an equivalent and proof irrelevant predicate [19]. Finite types come with a power operation: if X and Y are finite types, so is X^Y. We call $v : X^Y$ a **vector** and the type X^Y a **vector type**. The vector type X^Y represents all functions from Y to X up to extensionality. For a vector $v : X^Y$ we write v_y for the element of X to which y is mapped. To build a vector of type X^Y we write $(fy)_y$ where f is a function from Y to X.

Quantification over finite types preserves decidability. If a predicate p is decidable, then $\forall x.\ px$ and $\exists x.\ px$ are decidable, too.

Fact 2.1. *Let p be an decidable predicate on $\mathbb{N}$. Then the following is decidable:*

$$\exists n \leq k.\ pn \qquad \exists n < k.\ pn$$
$$\forall n \leq k.\ pn \qquad \forall n < k.\ pn$$

We say that a proposition p is **logically decidable** if $p \vee \neg p$. Because we are working in a constructive logic without excluded middle, not all propositions are logically decidable.

A type X is **countable** if there are functions $f : X \to \mathbb{N}$ and $g : \mathbb{N} \to X_\perp$ such that $\forall x.\ g(fx) = \mathsf{Some}\ x$. In particular $\mathbb{N}$ and all finite types are countable. Every countable type X admits **constructive choice**. This means that X has a choice operator. Let p be a decidable predicate on X. Then there is a function of type $(\exists x.\ px) \to X$ such that p is satisfied by the element of X given by the function.

For $\mathbb{N}$ there are specialized choice functions: For a decidable predicate p on $\mathbb{N}$ and a number n one can give a function $\mathsf{next} : (\exists m \geq n.\ pm) \to \mathbb{N}$ which computes the least position greater or equal to n at which p holds. If H is a proof of $\exists m \geq n.\ pm$, then

1. $p(\mathsf{next}\ H)$,
2. $n \leq \mathsf{next}\ H$, and
3. $\forall k.\ n \leq k < \mathsf{next}\ H \to \neg pk$.

If the existence of a proof for $\exists m \geq n.\ pm$ is clear from the context, we write $\mathsf{next}\ p\ n$ because the proof is only used to guarantee termination, but in constructive type theory the proof is the main argument.

2.2 Strings

An **alphabet** is a type and we use letters Σ or Γ for alphabets. The members of an alphabet are called **symbols**. We use letters a, b, c, and d for symbols. In this section alphabets may be finite or infinite if not stated differently. Later, we restrict ourselves to finite alphabets.

A **string** over Σ is a list over Σ. We denote the type of strings over Σ with Σ^* and the type of nonempty strings over Σ with Σ^+. We use letters x, y, u, and v for strings and write ε for the empty string. The length of a string is provided by $|x|$ and the n-th element of a string by $x[n]$ if $n < |x|$. The concatenation of two strings is denoted by $x \cdot y$ or by xy. We write $x[n..m)$ for the substring of x from n inclusively to m exclusively. Concatenating a string n times is written as x^n.

2.3 Sequences

An **ω-sequence** over an alphabet Σ is a function from $\mathbb{N}$ to Σ. We will use the term sequences if it is clear that we refer to ω-sequences. We denote the type of sequences with Σ^ω and use letters σ and τ for sequences. The n-th symbol of a sequence σ is simply σn.

The operation $x \cdot \sigma$ **prepends** the string x on the sequence σ:

$$\begin{aligned}(a \cdot \sigma)\ 0 &:= a\\ (a \cdot \sigma)\ (n+1) &:= \sigma\ n\\ \varepsilon \cdot \sigma &:= \sigma\\ (a :: x) \cdot \sigma &:= a \cdot (x \cdot \sigma)\end{aligned}$$

The operation $\sigma[n..]$ **drops** the first n positions of σ and is defined as

$$\begin{aligned}\sigma[0..] &:= \sigma\\ \sigma[n+1..]\ k &:= \sigma[n..](k+1)\end{aligned}$$

The operation $\sigma[0..n)$ yields the **prefix** of length n of σ :

$$\begin{aligned}\sigma[0..0) &:= \varepsilon\\ \sigma[0..n+1) &:= (\sigma\ 0) :: (\sigma[1..][0..n))\end{aligned}$$

The operation $\sigma[n..n+k)$ yields the string of length k starting at position n in σ.

$$\begin{aligned}\sigma[n..n+k) &:= \sigma[n..][0..k)\\ \sigma[n..m) &:= \sigma[n..n+(m-n))\end{aligned}$$

We use the second definition in the case that $m \geq n$ only (although it is defined always).

Fact 2.2. *The former operations behave as expected, e.g.*

$$\begin{aligned}(x \cdot \sigma)n &= x[n] && \text{for } n < |x|\\ (x \cdot \sigma)(|x| + n) &= \sigma n\\ \sigma[n..]m &= \sigma(n+m)\\ \sigma[n..m)[k] &= \sigma\ (n+k) && \text{for } k \leq (m-n)\end{aligned}$$

Definition 2.3 (Equivalence of Sequences)**.** *Two sequences are* ***equivalent*** *if they are pointwise equal.*

$$\sigma \equiv \tau := \forall n.\ \sigma n = \tau n$$

Equivalent sequences are not necessarily equal because sequences are functions.

Fact 2.4. *The operations above are compatible with sequence equivalence. If $\sigma \equiv \tau$, then*

$$\sigma[n..] \equiv \tau[n..] \qquad x \cdot \sigma \equiv x \cdot \tau \qquad \sigma[n..m) = \tau[n..m)$$

Fact 2.5. *Dropping cancels prepending and vice versa.*

$$(x\sigma)[|x|..] \equiv \sigma \qquad (\sigma[0..n)) \cdot (\sigma[n..]) \equiv \sigma$$

Sequences can be **composed** with other functions. For a sequence σ over Σ and a function f from Σ to Γ composition is defined as

$$\sigma \circ f := \lambda n.f(\sigma n).$$

The **product operation** $\otimes$ combines a sequence over Σ and another one over Γ to one sequence over $\Sigma \times \Gamma$.

$$\sigma \otimes \tau := \lambda n.(\sigma n, \tau n)$$

We generalize the product operation from two sequences to finitely many sequences: Let X be a finite type and f a function from X to Σ^ω. The operation $\bigotimes_X f$ yields the sequence over Σ^X by taking the product over all fx. $\bigotimes_X f$ is defined recursively over the list of elements of X. For a sequence σ over Σ^X the function π_x yields the sequence over Σ when choosing the entry for x at each position : $\pi_x \sigma n := (\sigma n)_x$.

Fact 2.6. *Let X be a finite type and f be a function from X to Σ^ω. Then $\pi_x(\bigotimes_X f) \equiv fx$.*

Lastly, we need to introduce a function @ yielding a sequence which is constant a apart from position n where it is b:

$$a@^n b := \lambda m.\mathsf{if}\ n = m\ \mathsf{then}\ b\ \mathsf{else}\ a$$

A function f from $\mathbb{N}$ to $\mathbb{N}$ is **strictly monotone** if $\forall n.\ fn < f(n+1)$. Strictly monotone sequences can be used to form subsequences: if f is strictly monotone, than $f \circ \sigma$ is the subsequence of σ containing only the positions given by f.

Fact 2.7. *If f and g are strictly monotone, then $f \circ g$ is strictly monotone.*

The following lemma is sometimes known as the finite pigeonhole principle and requires the alphabet to be finite.

Lemma 2.8. *For any finite alphabet Γ and for any string x with $|x| > |\Gamma|$ (or sequence σ) we can construct two indices $i < j$ such that $x[i] = x[j]$ (or $\sigma i = \sigma j$).*

Proof. Brute force searching for a duplicate symbol in $\sigma[0..k+1)$ must find such a symbol, because otherwise there are more than $|\Gamma|$ different symbols in Γ. □

2.3.1 ω-Concatenation and ω-Iteration

Let f be a function from $\mathbb{N}$ to Σ^+. The operation Πf yields the **ω-concatenation** of all strings in f to the sequence $f0 \cdot f1 \cdot f2 \cdot \ldots$. To define Πf we use the function $\Pi_\# f$ yielding pairs (i, j) such that the ω-concatenation of f can be obtained as $(fi)[j]$:

$$\begin{aligned}
\Pi_\# f\ 0 :=& (0,0)\\
\Pi_\# f\ (n+1) :=& \mathsf{let}\ (i,j) := \Pi_\# f\ n\ \mathsf{in}\\
&\quad \mathsf{if}\ j+1 = |fi|\ \mathsf{then}\ (i+1,0)\ \mathsf{else}\ (i,j+1)\\
\Pi f\ n :=& \mathsf{let}\ (i,j) := \Pi_\# f\ n\ \mathsf{in}\ (fi)[j]
\end{aligned}$$

Note that it is crucial that f is a function to Σ^+ and not to Σ^*. Otherwise one might not obtain a sequence in the second case.

Fact 2.9. *If $\forall n.\ |fn| = |gn|$, then $\Pi_\# f \equiv \Pi_\# f$.*

Fact 2.10. *If $f \equiv g$, then $\Pi f \equiv \Pi g$.*

We will write Πx_i instead of Πf when we do not need f explicitly but we ensure that the x_is are given by a function. You still may read Πx_i as $x_0 \cdot x_1 \cdot x_2 \cdot \ldots$ for intuition.

The **ω-iteration** of a nonempty string x, written x^ω, is the sequence repeating x:

$$x^\omega := \Pi(\lambda n.x)$$

Fact 2.11. *The ω-iteration of x is unaffected by prepending or iterating x.*

$$x^\omega \equiv xx^\omega \qquad x^\omega \equiv x^n x^\omega \qquad x^\omega \equiv (x^n)^\omega,\ \textit{for } n > 0$$

Fact 2.12. $(xy)^\omega \equiv x(yx)^\omega$

2.3.2 Sequences Satisfying a Predicate Infinitely Often

Let p be a predicate on Σ.

Definition 2.13. *A sequence σ satisfies p **infinitely often** if $\forall n.\ \exists m \geq n.\ p(\sigma m)$.*

A symbol a **occurs infinitely often** in a sequence σ if σ satisfies the predicate $\lambda b.a = b$ infinitely often.

Fact 2.14. *If σ satisfies p infinitely often, then both $\sigma[n..]$ and $x\sigma$ satisfy p infinitely often.*

A string x satisfies p (at least) once if there is an $n < |x|$ with $p(x[n])$.

Fact 2.15. *Let f be a function from $\mathbb{N}$ to Σ^+. The sequence Πf satisfies p infinitely often if and only if fn satisfies p once for infinitely many n.*

Corollary 2.16. *If x satisfies p once, then x^ω satisfies p infinitely often.*

2.3.3 ω-Filter

Let p be a decidable predicate on Σ and let σ satisfy p infinitely often. We want to define the sequence $\sigma\#p$ of all positions of σ satisfying p and the **ω-filter** $\sigma|p$ yielding the subsequence of σ at which p holds everywhere.

$$(\sigma\#p)\ 0 := \mathsf{next}\ p'\ 0$$
$$(\sigma\#p)\ (n+1) := \mathsf{next}\ p'\ (1 + (\sigma\#p)\ n)$$
$$\sigma|p := (\sigma\#p) \circ \sigma$$

where $p' := \lambda m.p(\sigma m)$. Recall that we can use $\mathsf{next}\ p'\ n$ from Section 2.1 to obtain the smallest position greater or equal to n at which p holds. This is possible because σ satisfies p infinitely often. So $\sigma\#p$ and $\sigma|p$ are only defined if σ satisfies p infinitely often, which ensures that the ω-filter yields a sequence.

Fact 2.17. *The function $\sigma\#p$ is strictly monotone.*

Fact 2.18. *The function $\sigma\#p$ selects all positions of σ at which p holds.*

Fact 2.19. *The predicate p holds at all positions in $\sigma|p$.*

Consequently, if $(\sigma\#p)\ n < k < (\sigma\#p)\ (n+1)$ then $\neg p(\sigma k)$.

2.4 Languages

A **string language** is a predicate on Σ^* and an **ω-language** is a predicate on Σ^ω. We write $x \in L$ for Lx, $\sigma \in L$ for $L\sigma$, $L \subseteq \Sigma^\omega$ ($L \subseteq \Sigma^*$) if L is an ω-language (string language), and $\{x|px\}$ for $\lambda x.px$. Disjunction yields union, conjunction yields intersection, and negation yields complementation. We call two languages **equivalent** and write $L_1 \equiv L_2$ if L_1 and L_2 contain the same sequences (or strings). Because our type theory is intensional, equivalent languages are not necessarily equal. An ω-language L is **extensional** if L is closed under sequence equivalence. Because all ω-languages occurring later will be extensional, we use extensionality of languages tacitly.

Let f be a function from Σ to Γ. The **image** of $L_\omega \subseteq \Sigma^\omega$ under f is the sequences language obtained by applying f pointwise to sequences of L_ω:

$$f(L_\omega) := \{\sigma | \exists \tau \in L_\omega.\ \sigma \equiv \tau \circ f\}$$

For $L_\omega \subseteq \Gamma^\omega$ we write $f^{-1}(L_\omega)$ for the **preimage** of L_ω under f:

$$f^{-1}(L_\omega) := \{\sigma | \sigma \circ f \in L_\omega\}$$

Remark 2.20. *When applying morphisms to ω-languages, one usually considers functions from Σ to Γ^+. Such morphisms could be defined using ω-concatenation but this would make things more complicated and the simpler case suffices for our purpose.*

For languages $L \subseteq \Sigma^*$ and $L_\omega \subseteq \Sigma^\omega$ the language $L \cdot L_\omega$ **prepends** L to L_ω.

$$L \cdot L_\omega := \{\sigma | \exists x \in L, \tau \in L_\omega.\ \sigma \equiv x\tau\}$$

We write LL_ω for $L \cdot L_\omega$ if not confusing.

The ω**-iteration** of the string language $L \subseteq \Sigma^*$ is the ω-language L^ω that contains the sequences obtained by concatenating infinitely many nonempty strings from L.

$$L^\omega := \{\sigma | \exists f.\ \sigma \equiv \Pi f \wedge \forall n.\ (fn) \in L\}$$

Note that this definition allows $\varepsilon \in L$ because f is required to be a function from $\mathbb{N}$ to Σ^+.

Fact 2.21. *If there is a strictly monotone function g such that $g0 = 0$ and $\sigma[gn..g(n+1)) \in L$ for all n, then $\sigma \in L^\omega$.*

Proof. Pick $fn := \sigma[gn..g(n+1))$. Because $g0 = 0$ we have $\sigma \equiv \Pi f$ and $fn \in L$ by assumption. Hence $\sigma \in L^\omega$. □

2.5 Ultimately Periodic Sequences

Sequences of the form xy^ω are called **ultimately periodic**, or short UP. Such sequences can be represented finitely by the strings x and y. We prove that UP sequences are closed under the operations $\circ$, $\otimes$, and @.

Fact 2.22. *Let f be a function from Σ to Γ. Then $xy^\omega \circ f$ is UP as well.*

Proof. As $xy^\omega \circ f \equiv (fx)(fy)^\omega$ where fx it the string obtained by applying f pointwise the the symbols in x. □

The closure of UP sequences under $\otimes$ is more involved because the strings may have different length.

Fact 2.23. *If $|x| \leq n$, then there are x' and y' such that $xy^\omega \equiv x'y'^\omega$ and $|x'| = n$.*

Proof. We pick

$$x' := x \cdot y^n[0..n - |x|)$$
$$y' := y^n[n - |x|..|y^n|) \cdot y^n[0..n - |x|).$$

$x'y'^\omega \equiv xy(y^n)^\omega$ by Fact 2.12. Then $xy(y^n)^\omega \equiv xy^\omega$ by Fact 2.11. Finally, $|x'| = |x| + n - |x| = n$. □

Fact 2.24. *If $|x| = |u|$, then $xy^\omega \otimes uv^\omega$ is UP.*

Proof. We show $xy^\omega \otimes uv^\omega \equiv (x \otimes u)(y^{|v|} \otimes v^{|y|})^\omega$, where $x \otimes u$ is the string obtained by creating pairs of $x[n]$ and $u[n]$. By Fact 2.11 $y^\omega \equiv (y^{|v|})^\omega$ and $v^\omega \equiv (v^{|y|})^\omega$. Hence $(y^{|v|} \otimes v^{|y|})^\omega \equiv y^\omega \otimes v^\omega$ and the claim follows. □

Lemma 2.25. *The sequence $xy^\omega \otimes uv^\omega$ is UP.*

Proof. By Fact 2.23 we can assume that $|x| = |u|$ (if $|x| \leq |u|$ apply Fact 2.23 with $n := |u|$ or vice versa). Then the claim follows by Fact 2.24. □

Fact 2.26. *The sequence $a@^n b$ is UP.*

Proof. As $a@^n b \equiv a^n b a^\omega$. □

Chapter 3

Ramseyan Properties

We introduce two propositions from Ramsey theory: the existence of Ramsey factorizations and Additive Ramsey. Both are provable classically, but not constructively. We will show that they are equivalent and independent in this chapter.

A pair $(\Gamma, +)$ is a **finite semigroup** if Γ is a finite type and $+$ is a binary and associative function on Γ. Let $(\Gamma, +)$ be a fixed but arbitrary finite semigroup. Symbols of Γ are called colors. A color a is called **idempotent** if $a + a = a$. For $x \in \Gamma^+$ we write $\sum x$ for the sum of all colors in x.

$$\sum x := x[0] + \cdots + x[|x| - 1]$$

Let $\sigma \in \Gamma^\omega$. A strictly monotone function g from $\mathbb{N}$ to $\mathbb{N}$ is called a **Ramseyan factorization** for σ if g partitions σ into pieces of equal sum (cf. Figure 3.1), where it may ignore a prefix of σ. That is

$$\forall i.\ \sum \sigma[g(0)..g(1)) = \sum \sigma[g(i)..g(i+1)).$$

g is called an **idempotent Ramseyan factorization**, if the color $\sum \sigma[g(0)..g(1))$ is idempotent.

Definition 3.1 (RF, Ramseyan Factorizations)**.** *RF is the following proposition: Every sequence over* Γ *admits a Ramseyan factorization. Formally*

$$\mathsf{RF} := \forall \sigma.\ \exists \textit{ strictly montone } g.\ \forall i.\ \sum \sigma[g0..g1) = \sum \sigma[gi..g(i+1)).$$

It is crucial that Ramseyan factorizations can ignore a finite prefix. For example the sequence ab^ω in the semigroup $(\{a, b\}, +)$ with $a + a = a$, $a + b = a$, $b + a = b$, and $b + b = b$ admits no Ramseyan factorization g with $g0 = 0$ because $\sum ab^n = a$ but $\sum b^{n+1} = b$.

For the sequence $a^n b^\omega$ the prefix a^n may be arbitrarily long but must be ignored by any Ramseyan factorization for the same reason as for ab^ω. So clearly a Ramseyan factorization cannot be computed (and RF not proven constructively). For special sequences the existence of Ramseyan factorizations can be proven, e.g. for UP sequences:

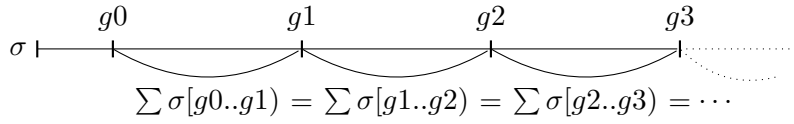


Figure 3.1: A Ramseyan factorization g for σ.

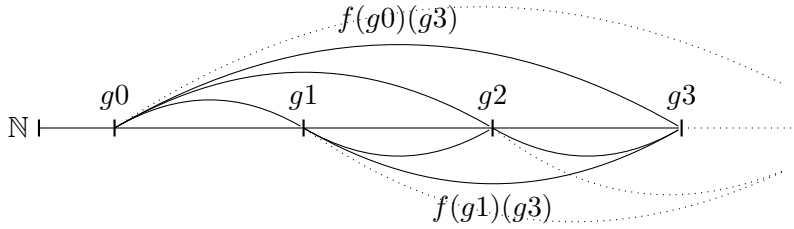


Figure 3.2: The strictly monotone and monochrome g given by AR. The colors $f(gi)(gj)$ with $i < j$ are always the same (for readability only two are marked in the figure).

Fact 3.2. *Any UP sequence xy^ω admits a Ramseyan factorization.*

Proof. The sequence $gn := |x| + n \cdot |y|$ is a Ramseyan factorization for xy^ω. Then for all i $\sum xy^\omega[gi..g(i+1)) = \sum y$ and hence $\sum xy^\omega[g0..g1)) = \sum xy^\omega[gi..g(i+1))$. □

Before introducing Additive Ramsey, we state *Ramsey's Theorem*: For any coloring f of unordered pairs of numbers with finitely many colors there are infinitely many numbers k_i such that f is constant on all pairs of distinct k_is.

Additive Ramsey restricts Ramsey's Theorem to special colorings f: the coloring f is restricted to use a finite semigroup as colors and to be additive. A function f of type $\mathbb{N} \to \mathbb{N} \to \Gamma$ is called **additive** if $(fij) + (fjk) = fik$ for $i < j < k$. The infinitely many numbers k_i are formalized by a strictly monotone function g yielding the k_i and the unordered pairs by only looking at pairs (i, j) with $i < j$. We say that g is **monochrome** if $f(gi)(gj)$ is constant for all $i < j$.

Definition 3.3 (AR, Additive Ramsey)**.** *AR is the following proposition: For every additive coloring f using colors Γ there is a strictly monotone and monochrome g (cf. Figure 3.2), meaning that $f(gi)(gj)$ is constant for all $i < j$. Formally*

$$\mathsf{AR} := \forall \text{ additive } f.\ \exists \text{ strictly monotone } g.\ \forall i < j.\ f(g0)(g1) = f(gi)(gj).$$

3.1 Equivalence of RF and AR

We show that RF is equivalent to AR. RF involves only unary functions while AR uses a binary function. AR is easier to use for proving things later. But when one needs to show that something implies AR, showing RF instead is simpler.

RF and AR are equivalent because additivity of a binary function f allows us to reduce f to a unary function $\sigma n = fn(n+1)$. Additivity implies $fij = \sum \sigma[i..j) = fi(i+1) + \cdots + f(j-1)j$ for $i < j$.

Lemma 3.4. *For any $a \in \Gamma$ one can construct a number k such that $k \cdot a$ is idempotent.*

Proof. For the sequence $\lambda n.2^n \cdot a$ one can construct two indices $i < i + k$ using Lemma 2.8 with $2^i \cdot a = 2^{i+k} \cdot a$. If $k = 1$ then $2^i \cdot a$ is idempotent: $2^i \cdot a + 2^i \cdot a = 2^{i+1} \cdot a = 2^i \cdot a$. If $k > 1$ then $(2^i \cdot (2^k - 1)) \cdot a$ is idempotent:

$$\begin{aligned}
&(2^i \cdot (2^k - 1)) \cdot a + (2^i \cdot (2^k - 1)) \cdot a \\
=&(2^{i+k}) \cdot a + (2^i \cdot (2^k - 2)) \cdot a \\
=&2^i \cdot a + (2^i \cdot (2^k - 2)) \cdot a \\
=&(2^i \cdot (2^k - 1)) \cdot a
\end{aligned}$$

□

Lemma 3.5. *A sequence admits a Ramseyan factorization if and only if it admits an idempotent Ramseyan factorization.*

Proof. The direction from idempotent Ramseyan factorizations to Ramseyan factorizations is trivial. To show the other direction, let g be a Ramseyan factorization of σ. By Lemma 3.4 there is a k such that $k \cdot \sum \sigma[g0..g1)$ is idempotent. Then the function $hn := g(k \cdot n)$ is an idempotent Ramseyan factorization for σ because

$$\begin{aligned}
&\sum \sigma[hi..h(i+1)) \\
=&\sum \sigma[g(k \cdot i)..g(k \cdot i + 1)) + \cdots + \sum \sigma[g(k \cdot i + k - 1)..g(k \cdot i + k)) \\
=&k \cdot \sum \sigma[g0..g1).
\end{aligned}$$

Hence, $\sum \sigma[hi..h(i+1)) = \sum \sigma[h0..h1)$ and $\sum \sigma[h0..h1)$ is idempotent. □

Lemma 3.6. *AR is equivalent to RF.*

Proof. AR ⇒ RF: We need to show that any σ admits a Ramseyan factorization. The function $fnm := \sum \sigma[n..m)$ is additive and by AR there is a strictly monotone and monochrome g. Then g is a Ramseyan factorization for σ because

$$\sum \sigma[gn..g(n+1)) = fn(n+1) = f01 = \sum \sigma[g0..g1).$$

RF $\Rightarrow$ AR: Let f be an additive function. By RF and Lemma 3.5 the sequence $\sigma n := fn(n+1)$ admits an idempotent Ramseyan factorization g. Then g is monochrome because for $i < j$

$$\begin{aligned} f(gi)(gj) &\overset{(1)}{=} \sum \sigma[gi..gj) \\ &\overset{(1)}{=} \sum \sigma[gi..g(i+1)) + \cdots + \sum \sigma[g(j-1)..gj) \\ &\overset{(2)}{=} \sum \sigma[g0..g1) + \cdots + \sum \sigma[g0..g1) \\ &\overset{(3)}{=} \sum \sigma[g0..g1) \overset{(1)}{=} f(g0)(g1). \end{aligned}$$

Steps (1) hold because f is additive, (2) holds because g is a Ramseyan factorization, and (3) holds because $\sum \sigma[0..1)$ is idempotent. □

3.2 Excluded Middle Implies RF

We show that excluded middle implies RF. We will split the proof into some carefully designed steps to reuse it for showing that SL implies RF. These steps make the usage of excluded middle explicit. The classical proof idea is given by Büchi [4], by Shelah [17], and in more details by Khoussaniov and Nerode [10]. Büchi and Shelah prove AR but proving RF in our formal setting is easier because one only needs to deal with unary functions.

Let $(\Gamma, +)$ be a finite semigroup and σ be a fixed sequence over Γ. We define the following relations:

$$\begin{aligned} i \simeq j \text{ (at } k) &:= i < k \wedge j < k \wedge \sum \sigma[i..k) = \sum \sigma[j..k) \\ i \simeq j &:= \exists k.\ i \simeq j \text{ (at } k) \end{aligned}$$

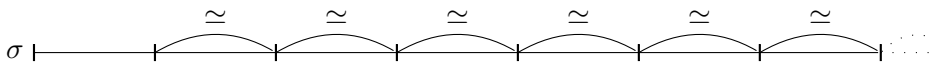
The later is an equivalence relation with at most $|\Gamma|$ many equivalence classes. Before we give the details of the proof, we outline the proof sketch:

By excluded middle there are infinitely many $\simeq$ equivalent positions (cf. Figure 3.3a). Given these infinitely many positions, there is a strictly monotone function g using constructive choice on $\mathbb{N}^2$ with the property that $g0 \simeq gi$ (at $g(i+1)$) (see Figure 3.3b). By excluded middle one color needs to occur infinitely often in the $\sum \sigma[g0..g(i+1))$s. The positions of g of that color are kept, all others are removed (cf. Figure 3.3c), and hence $\sum \sigma[g0..g1) = \sum \sigma[g0..g(i+1))$. Because $g(0) \simeq g(i)$ (at $g(i+1)$) we have

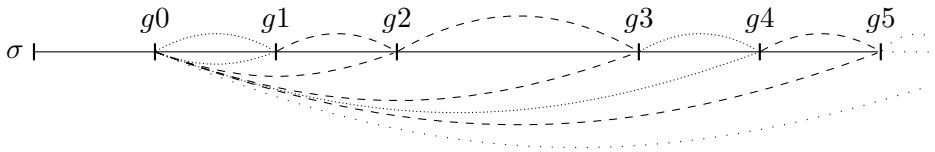
$$\sum \sigma[g(i)..g(i+1)) = \sum \sigma[g(0)..g(i+1)) = \sum \sigma[g(0)..g(1))$$

and the function g is indeed a Ramseyan factorization for σ.

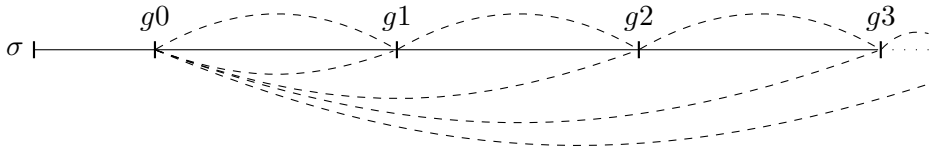
Note that excluded middle was needed twice in this proof. First to show that there are infinitely many $\simeq$ equivalent positions and second to show that one color occurs infinitely often in the $\sum \sigma[g0..g(i+1))$s. We factor



(a) Infinitely many $\simeq$ equivalent positions (not given by a function).



(b) Positions given by a strictly monotone function g such that $g0 \simeq gi$ (at $g(i+1)$). Equal sums $\sum \sigma[i..j)$ are indicated by equal line styles.



(c) Positions given by a strictly monotone function g such that $g0 \simeq gi$ (at $g(i+1)$) and $\sum \sigma[g0..g1) = \sum \sigma[g0..g(i+1))$. Only the dashed positions from before are kept. The function g is a Ramseyan factorization for σ because the $\sum \sigma[gi..g(i+1))$ are equal.

Figure 3.3: Construction of a Ramseyan factorization from infinitely many $\simeq$ equivalent positions.

these two points out of the proof to allow them to be proven differently and not by excluded middle. We make use of this to show that SL implies RF in Section 9.2.

Definition 3.7. *Let σ be an arbitrary sequence. The infinite pigeonhole principle is the proposition stating that there is a color a occurring infinitely often in σ. Formally:*

$$\mathsf{IPP} := \exists a.\ \forall i.\ \exists j \geq i.\ \sigma j = a.$$

The proposition $\mathsf{INF}_{\simeq}$ states that there is an i and for any j there is a $k > j$ such that $i \simeq k$. Intuitively this means that there are infinitely many $\simeq$ equivalent positions. Formally:

$$\mathsf{INF}_{\simeq} := \exists i.\ \forall j.\ \exists k > j.\ i \simeq k.$$

We write $\mathsf{DM}(\neg\mathsf{IPP})$ and $\mathsf{DM}(\neg\mathsf{INF}_{\simeq})$ for the proposition derived from $\neg\mathsf{IPP}$ and $\neg\mathsf{INF}_{\simeq}$ where the negation is pushed through the quantifiers with De Morgan laws.

We want to show that $\mathsf{DM}(\neg\mathsf{IPP})$ and $\mathsf{DM}(\neg\mathsf{INF}_{\simeq})$ are contradictory, hence that $\mathsf{IPP} \vee \mathsf{DM}(\neg\mathsf{IPP})$ implies IPP, that $\mathsf{INF}_{\simeq} \vee \mathsf{DM}(\neg\mathsf{INF}_{\simeq})$ implies $\mathsf{INF}_{\simeq}$, and finally that IPP and $\mathsf{INF}_{\simeq}$ imply RF. Because $\mathsf{IPP} \vee \mathsf{DM}(\neg\mathsf{IPP})$ and $\mathsf{INF}_{\simeq} \vee \mathsf{DM}(\neg\mathsf{INF}_{\simeq})$ follow directly from excluded middle, excluded middle implies RF. We begin with facts of the $\simeq$ equivalence.

Fact 3.8. *$i \simeq i$ (at $i + 1$).*

Fact 3.9. *$i \simeq j$ (at k) implies $i \simeq j$ (at n) for all $n \geq k$.*

Lemma 3.10. *The equivalence relation $\simeq$ has at most $|\Gamma|$ many equivalence classes.*

Proof. Let $n > |\Gamma|$ and i_1 to i_n be different positions. Then there are different j and k such that $\sum \sigma[i_j..i_n + 1) = \sum \sigma[i_k..i_n + 1)$, hence $i_j \simeq i_k$ and $\simeq$ cannot have more than $|\Gamma|$ many equivalence classes. □

Lemma 3.11. $\mathsf{INF}_{\simeq} \vee \mathsf{DM}(\neg\mathsf{INF}_{\simeq})$ *implies* $\mathsf{INF}_{\simeq}$.

Proof. If $\mathsf{INF}_{\simeq}$ holds we are done. Otherwise we have (after applying De Morgan laws to $\neg\mathsf{INF}_{\simeq}$)

$$\forall i.\ \exists j.\ \forall k > j.\ i \not\simeq k.$$

We show that $\simeq$ has infinitely many equivalence classes contradicting Lemma 3.10. Therefore we show that there are $m_1, \ldots, m_n$ and a position j for every n such that $m_i \not\simeq m_k$ for $i \neq k$ and $m_i \not\simeq k$ for any $k > j$. The position j is required to prove the claim by induction on n. For $n = 0$ the

claim is vacuously true for an arbitrary j. Assume there are $m_1, \ldots, m_n$ and a j satisfying the condition. Then it suffices to give an additional m_{n+1} and a new j'. Pick $m_{n+1} := j+1$. From above there is a $j' > j+1 = m_{n+1}$ such that $m_{n+1} \not\simeq k$ for all $k > j'$. $j' > j+1 = m_{n+1}$ implies $m_i \not\simeq m_{n+1}$ for $i \leq n$. Finally, $m_i \not\simeq k$ for $i \leq n$ and $k > j' > j$ by induction hypothesis. □

Lemma 3.12. *Given* $\mathsf{INF}_{\simeq}$*, there is a strictly monotone* g *such that* $g0 \simeq gi$ *(at* $g(i+1)$*) for all* i *(cf. Figure 3.3b).*

Proof. By $\mathsf{INF}_{\simeq}$ there is an i such that

$$\forall j.\ \exists k > j.\ i \simeq k \leftrightarrow \forall j.\ \exists km.\ m > k > j \wedge i \simeq k \text{ (at } m).$$

The equivalence follows by the definition of $\simeq$. Note that $m > k > j \wedge i \simeq k$ (at m) is decidable and that $\mathbb{N}^2$ is a countable type. Hence, we have constructive choice for the existential quantifier on the right and can construct a $\hat{k}(j)$ and $\hat{m}(j)$ for all j.

We want to show that there is a sequence of pairs (k_n, m_n) such that $k_0 \simeq k_n$ (at m_n) and $k_n < m_n < k_{n+1}$ for all n. The sequence is established recursively as:

$$\begin{aligned} k_0 &:= i & m_0 &:= i+1 \\ k_{n+1} &:= \hat{k}(m_n) & m_{n+1} &:= \hat{m}(m_n) \end{aligned}$$

The condition $k_n < m_n < k_{n+1}$ is easy to see because $j < \hat{k}(j) < \hat{m}(j)$ for all j. The other condition that $k_0 \simeq k_n$ (at m_n) is easy, too, because $k_0 = i$ and $i \simeq \hat{k}(j)$ (at $\hat{m}(j)$) for all j.

From $m_n < k_{n+1}$ follows $k_0 \simeq k_n$ (at k_{n+1}) by Fact 3.9. Hence, the function $gn := k_n$ is the desired one. □

Lemma 3.13. $\mathsf{IPP} \vee \mathsf{DM}(\neg\mathsf{IPP})$ *implies* IPP.

Proof. If IPP holds we are done. Otherwise

$$\forall a.\ \exists i.\ \forall j \geq i.\ \sigma j = a.$$

So there is a j_a for every a where a does not occur in σ after j_a. Then after $\max_a j_a$ no color of Γ occurs in σ which is a contradiction. □

Lemma 3.14. *Given* $\mathsf{INF}_{\simeq}$ *and* IPP*, there is a strictly monotone* g *such that*

$$\begin{gathered} g0 \simeq gi \text{ (at } g(i+1)) \text{ and} \\ \sum \sigma[g0..g1) = \sum \sigma[g0..g(i+1)) \end{gathered}$$

for all i *(cf. Figure 3.3c).*

Proof. By Lemma 3.12 there is an f satisfying the first condition. By IPP there is a color a occurring infinitely often in the sequence $\tau := \lambda n. \sum \sigma[f0..fn)$. Recall that $\tau\#a$ yields the strictly monotone sequence selecting all positions where a occurs in τ. We define $h0 = 0$ and $h(n+1) = (\tau\#a)(n+1)$. Clearly, h is strictly monotone. The function $g := h \circ f$ is the desired function. First, $g0 \simeq gi$ (at $g(i+1)$) by Fact 3.9 because $g0 = f0$, $gi = fj$ and $g(i+1) = fk$ for some j and k with $j < k$, and $f0 \simeq fj$ (at $f(j+1)$). Second, $\sum \sigma[g0..g1) = \sum \sigma[f0..(\tau\#a)1) = a = \sum \sigma[f0..(\tau\#a)(i+1)) = \sum \sigma[g0..g(i+1))$. □

Lemma 3.15. *Given* $\mathsf{INF}_{\simeq}$ *and* IPP*, the sequence* σ *admits a Ramseyan factorization.*

Proof. Let g be the strictly monotone function from Lemma 3.14. Then g is a Ramseyan factorization for σ. Lemma 3.14 implies $\sum \sigma[gi..g(i+1)) = \sum \sigma[g0..g(i+1))$ and $\sum \sigma[g0..g(i+1)) = \sum \sigma[g0..g1)$. □

Note that g is actually an idempotent Ramseyan factorization because $\sum \sigma[g0..g1) = \sum \sigma[g0..g2) = \sum \sigma[g0..g1) + \sum \sigma[g1..g2) = \sum \sigma[g0..g1) + \sum \sigma[g0..g2) = \sum \sigma[g0..g1) + \sum \sigma[g0..g1)$.

Corollary 3.16. *Excluded middle implies RF.*

Proof. Follows from Lemmas 3.15, 3.11, 3.13 and excluded middle because σ was arbitrary. □

3.3 Independence of RF

We want to prove that RF is independent in constructive type theory, meaning that it cannot be derived but consistently be assumed. We already know that excluded middle implies RF and we are going to show that RF implies Markov's principle. Excluded middle is independent. Coquand and Mannaa [7] showed that Markov's principle in independent in constructive type theory, too. As excluded middle and Markov's principle are independent, RF must be independent as well.

Definition 3.17 (Markov's principle)**.** *Markov's principle states that if a sequence* σ *over* $\mathbb{B}$ *is not constantly* true*, then there is an* n *such that* $\sigma n =$ false*. Formally*

$$\forall \sigma.\ \neg(\forall n.\ \sigma n = \mathsf{true}) \to \exists n.\ \sigma = \mathsf{false}.$$

Lemma 3.18. *RF implies Markov's principle.*

Proof. Let σ be a sequence over $\mathbb{B}$ which is not constantly true. We need to show that there is an n with $\sigma n =$ false. By RF σ admits a Ramseyan factorization g in the finite semigroup $(\mathbb{B}, \&)$ where & is conjunction on $\mathbb{B}$.

If $\&\sigma[g0..g1) = \mathsf{true}$ and hence $\&\sigma[gi..g(i+1)) = \mathsf{true}$ for all i, σ is constantly true after $g0$. It is decidable whether there is an $n < g0$ with $\sigma n = \mathsf{false}$. If such an n exists, the claim is proven. Otherwise σ is constantly true contradicting the assumption.

If $\&\sigma[g0..g1) = \mathsf{false}$ then there must be an n such that $g0 \leq n < g1$ and $\sigma n = \mathsf{false}$ and the claim is proven. □

Theorem 3.19. *RF is independent in constructive type theory.*

Proof. If RF was derivable, then Markov's principle would be derivable by Lemma 3.18. If RF was inconsistent, then excluded middle would be inconsistent by Corollary 3.16. Because Markov's principle and excluded middle are independent, RF is independent as well. □

Chapter 4

NFAs

Nondeterministic finite automata (NFAs) can be used to run on strings and sequences. Strings and sequences need different acceptance criteria because sequences do not have a last symbol. We call the acceptance for strings regular acceptance, which is the usual acceptance on strings yielding regular languages. For ω-sequences we use Büchi acceptance. We use NFAs instead of DFAs because Büchi acceptance on DFAs is strictly weaker than on NFAs. Usually, NFAs with Büchi acceptance are called Büchi automata, but the underlying automata are the same as for NFAs, there is only a different acceptance criterion. In this chapter we introduce NFAs, the concept of runs of NFAs on strings and sequences, and regular and Büchi acceptance. We give some selected properties for NFAs with regular acceptance, which are later needed for properties of Büchi acceptance. Last, we show that emptiness of the sequence language of NFAs is decidable.

Formally, an **NFA** over a finite alphabet Σ is a structure $(Q, I, F, \to)$ where

- Q is a finite type of states,
- I is a decidable predicate on Q identifying the initial states,
- F is a decidable predicate on Q identifying the final states, and
- $\to$ is a decidable transition relation of type $Q \to \Sigma \to Q \to \mathbb{P}$.

For a state q we write $q \in I$ for Iq, $q \in F$ for Fq, and $q \xrightarrow{a} p$ for $\to qap$. A **finite run** of an NFA is a string over Q. An **infinite run** is an ω-sequence over Q. We denote finite runs with r or s and infinite runs with ϱ or ξ. If clear from the context whether a run is infinite or finite, we only say run. Finite and infinite runs are introduced together to make definitions and lemmas uniform, which simplifies combining finite and infinite runs.

Definition 4.1 (Valid Runs)**.** *A finite run r of an NFA $\mathcal{A}$ is **valid on a string** x if $|r| = |x| + 1$ and $r[n] \xrightarrow{x[n]} r[n+1]$ for all $n < |x|$. An infinite run ϱ is **valid on a sequence** σ if $\varrho n \xrightarrow{\sigma n} \varrho(n+1)$ for all n.*

We establish some facts on manipulating valid runs. Let $\mathcal{A}$ be an arbitrary NFA.

Fact 4.2. *If $\varrho \equiv \xi$, $\sigma \equiv \tau$, and ϱ is valid on σ, then ξ is valid on τ.*

Fact 4.3. *If ϱ is valid on σ, then $\varrho[n..]$ is valid on $\sigma[n..]$ and $\varrho[n..m+1)$ is valid on $\sigma[n..m)$ for $m > n$.*

Fact 4.4. *If r is valid on x, ϱ is valid on σ, and $r[|x|] = \varrho 0$, then $r[0..|x|) \cdot \varrho$ is valid on $x\sigma$.*

Lemma 4.5. *Let $r_i \in Q^+$, $x_i \in \Sigma^+$ for all i, and both the r_is and x_is be computable by a function. If $r_i(r_{i+1}[0])$ is valid on x_i for all i, then Πr_i is valid on Πx_i.*

Proof. We need to show $(\Pi r_i)n \overset{(\Pi x_i)n}{\to} (\Pi r_i)(n+1)$. Note that $|x_i| = |r_i|$ for all i. Hence, we have to show $r_i[j] \overset{x_i[j]}{\to} r_i[j+1]$ where $j+1 < |r_i|$ and $r_i[|r_i|-1] \overset{x_i[|r_i|-1]}{\to} r_{i+1}[0]$. This is precisely that $r_i(r_{i+1}[0])$ is valid on x_i. □

Corollary 4.6. *If qrq is valid on x, then $(qr)^\omega$ is valid on x^ω.*

Fact 4.7. *It is decidable whether r is valid on x.*

4.1 Reachability in NFAs

We define reachability in NFAs and show that reachability is decidable.

Definition 4.8. *A run r and a string x form a **path** from q to p if r is valid on x, $r[0] = q$, and $r[|r|-1] = p$.*

Definition 4.9. *A state p is **reachable** from a state q if there are an r and a nonempty x such that r and x form a path from q to p.*

Because x is required to be nonempty, a state is not necessarily reachable from itself but only when there is a loop. This saves us case distinctions later on. We define reachability using strings and finite runs and not inductively, because then the strings and finite runs can be used to build sequences and infinite runs.

Lemma 4.10. *The state q is reachable from p if and only if q is reachable from p on a run of length at most $|Q|+1$.*

Proof. Assume that r and x form a path from q to p. If $|r| \leq |Q|+1$ we are done. Otherwise one can construct by Lemma 2.8 two positions $i < j < |r|-1$ such that $r[i] = r[j]$. Then $r[0..i)r[j..|r|)$ is valid on $x[0..i)x[j..|x|)$. The new string is nonempty because $j < |r|-1$. Because the length of the run and the string decrease by at least one, we can repeat this argument until $|r| \leq |Q|+1$ ($|r|$ many times suffices). □

It is necessary to allow runs of length $|Q| + 1$ because a state q may be reachable from itself only when visiting every other state.

Lemma 4.11. *Reachability for NFAs is decidable.*

Proof. By Lemma 4.10 one needs to look for runs of length at most $|Q| + 1$ only. Hence, there are only finitely many runs and strings to consider. Because being valid is decidable by Fact 4.7, reachability is decidable, too. □

4.2 Regular and Büchi Acceptance

We define regular and Büchi acceptance for NFAs. Let $\mathcal{A} = (Q, I, F, \to)$ be an NFA over Σ.

Definition 4.12 (Regular Acceptance)**.** *A finite run r is **initial** if $r[0] \in I$ and **final** if $r[|r| - 1] \in F$. A run is **accepting** on x if it is valid on x, initial, and final. An NFA **accepts** x if there is an accepting run on x. The **regular language** of an NFA is the language of accepted strings:*

$$\mathscr{L}_R(\mathcal{A}) := \{x \in \Sigma^* | \mathcal{A} \text{ accepts } x\}$$
$$\mathscr{L}_R^+(\mathcal{A}) := \{x \in \Sigma^+ | x \in \mathscr{L}_R(\mathcal{A})\}$$

The language of nonempty strings will be useful later, e.g. when ω-iterating the regular language of an NFA. Again, we do not use an inductive definition for string acceptance.

Definition 4.13 (Büchi Acceptance)**.** *An infinite run ϱ of an NFA $\mathcal{A}$ is **initial** if $\varrho 0 \in I$ and **final** if there are infinitely many final states in ϱ. Formally, this means that ϱ satisfies F infinitely often (recall that F is a predicate). An infinite run is **accepting** on σ if it is valid on σ, initial, and final. The NFA $\mathcal{A}$ **accepts** σ if there is an accepting run of σ. The **(Büchi) language** of an NFA is the language of accepted sequences.*

$$\mathscr{L}_B(\mathcal{A}) := \{\sigma \in \Sigma^\omega | \mathcal{A} \text{ accepts } \sigma\}$$

An NFA $\mathcal{A}$ **accepts** an ω-language (a string language) L if $\mathscr{L}_B(\mathcal{A}) \equiv L$ ($\mathscr{L}_R(\mathcal{A}) \equiv L$). If the sequence (string) is clear from the context, we say that a run is accepting. We speak of a Büchi automaton to clearify that we refer to NFAs and are interested in Büchi acceptance. If not confusing, we say language of an NFA (or Büchi automaton) to refer to the language of accepted sequences.

Lemma 4.14 (Constructive Choice for Accepting Finite Runs)**.** *For any $x \in \mathscr{L}_R(\mathcal{A})$ one can construct an accepting run.*

Proof. The finite run needs to be of length $|x| + 1$, hence there are only finitely many runs to check. There exists one, because $x \in \mathscr{L}_R(\mathcal{A})$. □

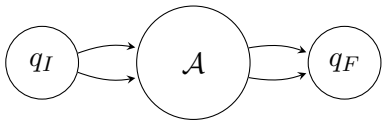


Figure 4.1: A normalized NFA.

4.3 Properties of NFAs With Regular Acceptance

An NFA is **normalized** if it has an unique initial state q_I and an unique final state q_F different from q_I such that no transition enters q_I and no transition leaves q_F (cf. Figure 4.1).

Lemma 4.15 (Normalization)**.** *For an NFA $\mathcal{A}$ one can construct a normalized NFA $\mathcal{A}'$ with $\mathscr{L}_R^+(\mathcal{A}) \equiv \mathscr{L}_R(\mathcal{A}')$.*

Proof. Let $\mathcal{A} = (Q, I, F, \to)$ and $\mathcal{A}' := (Q + \{1,2\}, \{1\}, \{2\}, \rightsquigarrow)$ where $\rightsquigarrow$ is defined as follows:

$$1 \overset{a}{\rightsquigarrow} p := \exists q \in I.\ q \overset{a}{\to} p$$
$$q \overset{a}{\rightsquigarrow} p := q \overset{a}{\to} p$$
$$q \overset{a}{\rightsquigarrow} 2 := \exists p \in F.\ q \overset{a}{\to} p$$
$$1 \overset{a}{\rightsquigarrow} 2 := \exists q \in I, p \in F.\ q \overset{a}{\to} p$$

Clearly no transition enters 1 and no transition leaves 2.

To show $\mathscr{L}_R^+(\mathcal{A}) \subseteq \mathscr{L}_R(\mathcal{A}')$, let $x \in \mathscr{L}_R^+(\mathcal{A})$, and r be an accepting run for x. Then $1 \cdot r[1..|r|-1) \cdot 2$ is an accepting run for $\mathcal{A}'$.

To show $\mathscr{L}_R(\mathcal{A}') \subseteq \mathscr{L}_R^+(\mathcal{A})$, let $x \in \mathscr{L}_R(\mathcal{A}')$ and r be an accepting run. Then $r = 1 \cdot r[1..|r|-1) \cdot 2$. By case analysis on $|x| = 1$ there are $q \in I$ and $p \in F$ such that $q \cdot r[1..|r|-1) \cdot p$ is an accepting run for $\mathcal{A}$. □

Doczkal and Smolka [8] give a characterization of regular languages with so-called classifers. Classifiers correspond to Myhill-Nerode relations in constructive type theory. A function f from Σ^* to a finite type X is called a **classifier** if f is right congruent: $f(x) = f(y)$ implies $f(xa) = f(ya)$ for all x, y, and a. A classifier f is a classifier for a language L if f **refines** L: $x \in L \leftrightarrow y \in L$ whenever $fx = fy$.

Lemma 4.16 ([8])**.** *If a string language L is decidable and there is a classifier for L, then one can construct an NFA accepting L.*

4.4 Decidability of Emptiness of Büchi Automata

We show that emptiness of languages of Büchi automata is decidable. It suffices to decide the existence of final loops.

Definition 4.17. *An NFA $\mathcal{A}$ has a **final loop** if there is an initial state $q \in I$ and a final state $p \in F$ such that p is reachable from q and p is reachable from p.*

Lemma 4.18. *If $\sigma \in \mathscr{L}_B(\mathcal{A})$, then $\mathcal{A}$ has a final loop.*

Proof. Let ϱ be an accepting run for σ. Then by Lemma 2.8 there are $0 < i < j$ such that $(\varrho|F)i = (\varrho|F)j$ (recall that $\varrho|F$ is the subsequence of ϱ of all final states), which means that $(\varrho|F)i$ is reachable from itself and from $\varrho 0$. Hence, $\mathcal{A}$ has a final loop. □

Lemma 4.19. *If $\mathcal{A}$ has a final loop, then there is an $xy^\omega \in \mathscr{L}_B(\mathcal{A})$.*

Proof. There are $q \in I$ and $p \in F$ such that p is reachable both from q and p. Then there are x, y, r, and s where qrp is valid on x and psp is valid on y, and hence x and y are nonempty. By Corollary 4.6 and Fact 4.4 $qr(ps)^\omega$ is valid on xy^ω and by Corollary 2.16 and Fact 2.14 $qr(ps)^\omega$ is final because $p \in F$. Because $q \in I$, $qr(ps)^\omega$ is initial and hence $xy^\omega \in \mathscr{L}_B(\mathcal{A})$. □

For this lemma it was crucial that reachability requires strings to be nonempty, because otherwise xy^ω may not be a sequence.

Lemma 4.20. *It is decidable whether $\mathcal{A}$ has a final loop.*

Proof. By Lemma 4.11 reachability is decidable and because quantifiers over finite types preserve decidability, it is decidable whether $\mathcal{A}$ has a final loop. □

Lemma 4.21 (Decidability of Emptiness)**.** *It is decide whether $\mathscr{L}_B(\mathcal{A}) \equiv \emptyset$ or whether there is a $xy^\omega \in \mathscr{L}_B(\mathcal{A})$.*

Proof. By Lemma 4.20 it is decidable whether $\mathcal{A}$ has a final loop. If $\mathcal{A}$ has a final loop, there is an $xy^\omega \in \mathscr{L}_B(\mathcal{A})$ by Lemma 4.19. If $\mathcal{A}$ has no final loop, we need to show $\sigma \notin \mathscr{L}_B(\mathcal{A})$ for all σ. Assume $\sigma \in \mathscr{L}_B(\mathcal{A})$, then by Lemma 4.18 $\mathcal{A}$ has a final loop, which is a contradiction. □

Corollary 4.22. *If an NFA accepts some sequence, then is accepts an UP sequence.*

Chapter 5

Basic Operations on Büchi Automata

In this chapter we study closure operations of Büchi automata. Complementation is covered in Chapter 6 because it is more complicated. It is important to have operations for the closure properties, because S1S formulas should be translated into Büchi automata. To decide satisfiability of formulas, the Büchi automaton must be computable. We follow the classical constructions. For the correctness proofs we focus on difficulties in constructive logic, e.g. the use of constructive choice.

Lemma 5.1 (Closure under Preimage)**.** *Let f be a function from Σ to Γ and $\mathcal{A}$ an NFA over Γ. Then one can construct an NFA $f^{-1}(\mathcal{A})$ over Σ which accepts the preimage of $\mathscr{L}_B(\mathcal{A})$ under f:*

$$\mathscr{L}_B(f^{-1}(\mathcal{A})) \equiv f^{-1}(\mathscr{L}_B(\mathcal{A}))$$

Proof. Let $\mathcal{A} = (Q, I, F, \to)$ and $f^{-1}(\mathcal{A}) := (Q, I, F, \rightsquigarrow)$ where $q \overset{a}{\rightsquigarrow} p := q \overset{f(a)}{\to} p$. Verification is straight forward. □

Lemma 5.2 (Closure under Image)**.** *Let f be a function from Σ to Γ and $\mathcal{A}$ an NFA over Σ. Then one can construct an NFA $f(\mathcal{A})$ over Γ accepting the image of $\mathscr{L}_B(\mathcal{A})$ under f:*

$$\mathscr{L}_B(f(\mathcal{A})) \equiv f(\mathscr{L}_B(\mathcal{A}))$$

Proof. Let $\mathcal{A} = (Q, I, F, \to)$ and $f(\mathcal{A}) := (Q, I, F, \rightsquigarrow)$ where

$$q \overset{a}{\rightsquigarrow} p := \exists b.\ fb = a \wedge q \overset{b}{\to} p.$$

$f(\mathscr{L}_B(\mathcal{A})) \subseteq \mathscr{L}_B(f(\mathcal{A}))$: Let $\sigma \in f(\mathscr{L}_B(\mathcal{A}))$. Then there is a $\tau \in \mathscr{L}_B(\mathcal{A})$ with $\sigma \equiv \tau \circ f$. Let ϱ be an accepting run for $\mathcal{A}$ on τ. Then ϱ is an accepting run for $f(\mathcal{A})$ on σ.

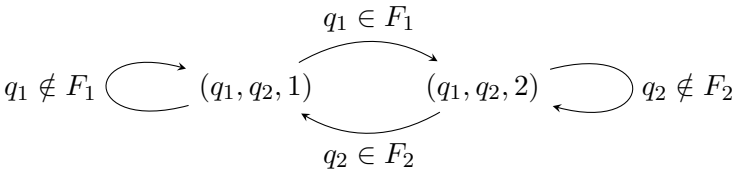


Figure 5.1: The usage of the additional component in the transition relation of the intersection Büchi automaton.

$\mathscr{L}_B(f(\mathcal{A})) \subseteq f(\mathscr{L}_B(\mathcal{A}))$: Let $\sigma \in \mathscr{L}_B(f(\mathcal{A}))$ and ϱ be an accepting run. Then for all n there is a b with $fb = \sigma n$. By constructive choice one obtains such a $b(n)$. Then $(\lambda n.b(n)) \in \mathscr{L}_B(\mathcal{A})$ because ϱ is an accepting run and $\sigma \in f(\mathscr{L}_B(\mathcal{A}))$ because $\lambda n.b(n) \equiv \sigma \circ f$. □

Lemma 5.3 (Closure under Union)**.** *For two NFAs $\mathcal{A}_1$ and $\mathcal{A}_2$ one can construct $\mathcal{A}_1 \cup \mathcal{A}_2$ accepting the union of the languages of $\mathcal{A}_1$ and $\mathcal{A}_2$.*

$$\mathscr{L}_B(\mathcal{A}_1 \cup \mathcal{A}_2) \equiv \mathscr{L}_B(\mathcal{A}_1) \cup \mathscr{L}_B(\mathcal{A}_2)$$

Proof. Let $\mathcal{A}_1 = (Q_1, I_1, F_1, \to_1)$ and $\mathcal{A}_2 = (Q_2, I_2, F_2, \to_2)$. Define $\mathcal{A}_1 \cup \mathcal{A}_2 := (Q_1 + Q_2, I_1 \cup I_2, F_1 \cup F_2, \to_1 \cup \to_2)$. The constructors of the sum type are used implicitly. Verification is straightforward. □

Lemma 5.4 (Closure under Intersection)**.** *For every $\mathcal{A}_1$ and $\mathcal{A}_2$ one can construct $\mathcal{A}_1 \cap \mathcal{A}_2$ accepting the intersection of the languages of $\mathcal{A}_1$ and $\mathcal{A}_2$.*

$$\mathscr{L}_B(\mathcal{A}_1 \cap \mathcal{A}_2) \equiv \mathscr{L}_B(\mathcal{A}_1) \cap \mathscr{L}_B(\mathcal{A}_2)$$

Proof. Let $\mathcal{A}_1 = (Q_1, I_1, F_1, \to_1)$, $\mathcal{A}_2 = (Q_2, I_2, F_2, \to_2)$ and $\mathcal{A}_1 \cap \mathcal{A}_2 := (Q_1 \times Q_2 \times \{1,2\}, I_1 \times I_2 \times \{1,2\}, Q_1 \times F_2 \times \{2\}, \rightsquigarrow)$ where

$$\begin{aligned}
(q_1, q_2, 1) \overset{a}{\rightsquigarrow} (p_1, p_2, 1) &:= q_1 \overset{a}{\to}_1 p_1 \wedge q_2 \overset{a}{\to}_2 p_2 \wedge q_1 \notin F_1 \\
(q_1, q_2, 1) \overset{a}{\rightsquigarrow} (p_1, p_2, 2) &:= q_1 \overset{a}{\to}_1 p_1 \wedge q_2 \overset{a}{\to}_2 p_2 \wedge q_1 \in F_1 \\
(q_1, q_2, 2) \overset{a}{\rightsquigarrow} (p_1, p_2, 1) &:= q_1 \overset{a}{\to}_1 p_1 \wedge q_2 \overset{a}{\to}_2 p_2 \wedge q_2 \in F_2 \\
(q_1, q_2, 2) \overset{a}{\rightsquigarrow} (p_1, p_2, 2) &:= q_1 \overset{a}{\to}_1 p_1 \wedge q_2 \overset{a}{\to}_2 p_2 \wedge q_2 \notin F_2.
\end{aligned}$$

The automaton runs $\mathcal{A}_1$ and $\mathcal{A}_2$ in parallel. To assert that both $\mathcal{A}_1$ and $\mathcal{A}_2$ visit a final state infinitely often, the intersection NFA uses the additional component in its states to assert that it infinitely often visits first a final state of $\mathcal{A}_1$ and then later one of $\mathcal{A}_2$. The NFA stores 1 as last component of its current state until it visits a final state of $\mathcal{A}_1$, then swaps to 2, and stores 2 until it visits a final state of $\mathcal{A}_2$ (cf. Figure 5.1).

$\mathscr{L}_B(\mathcal{A}_1 \cap \mathcal{A}_2) \subseteq \mathscr{L}_B(\mathcal{A}_1) \cap \mathscr{L}_B(\mathcal{A}_2)$: Let ϱ be an accepting run for $\sigma \in \mathcal{A}_1 \cap \mathcal{A}_2$. We first show $\sigma \in \mathscr{L}_B(\mathcal{A}_2)$. The run $\varrho \circ \pi_2$ is valid and initial on σ for $\mathcal{A}_2$ by definition of $\rightsquigarrow$. It is final because whenever ϱ is final $\varrho \circ \pi_2$ is final,

too. It remains to show $\sigma \in \mathscr{L}_B(\mathcal{A}_1)$. The run $\varrho \circ \pi_1$ is valid and initial on σ for $\mathcal{A}_1$. It is also final, because between two final states of $\mathcal{A}_1 \cap \mathcal{A}_2$ there must be a final state of $\mathcal{A}_1$ somewhere. This can be proven by contradiction because it is decidable whether between two positions there is a final state of $\mathcal{A}_1$.

$\mathscr{L}_B(\mathcal{A}_1) \cap \mathscr{L}_B(\mathcal{A}_2) \subseteq \mathscr{L}_B(\mathcal{A}_1 \cap \mathcal{A}_2)$. Let $\sigma \in \mathscr{L}_B(\mathcal{A}_1) \cap \mathscr{L}_B(\mathcal{A}_2)$. Then there are an accepting run ϱ for $\mathcal{A}_1$ and an accepting run ξ for $\mathcal{A}_2$. Note that $\rightsquigarrow$ is deterministic in its third component given ϱ and ξ as first and second component when choosing 1 for the initial third component. Hence, a run for $\mathcal{A}_1 \cap \mathcal{A}_2$ can be constructed using a recursive function following the transition relation. This gives an accepting run for $\mathcal{A}_1 \cap \mathcal{A}_2$. It is initial and valid by construction. It is final because the third component must switch infinitely often between 1 and 2 because both $\mathcal{A}_1$ and $\mathcal{A}_2$ visit final states infinitely often. □

Lemma 5.5 (Closure under Prepending NFAs)**.** *For every $\mathcal{A}_1$ and $\mathcal{A}_2$ one can construct an NFA $\mathcal{A}_1 \cdot \mathcal{A}_2$ such that*

$$\mathscr{L}_B(\mathcal{A}_1 \cdot \mathcal{A}_2) \equiv \mathscr{L}_R(\mathcal{A}_1) \cdot \mathscr{L}_B(\mathcal{A}_2).$$

Proof. Let $\mathcal{A}_1 = (Q_1, I_1, F_1, \rightarrow_1)$, $\mathcal{A}_2 = (Q_2, I_2, F_2, \rightarrow_2)$, and $\mathcal{A}_1 \cdot \mathcal{A}_2 := (Q_1 + Q_2, I, F_2, \rightsquigarrow)$ where

$$\begin{aligned}
q \overset{a}{\rightsquigarrow} p &:= q \overset{a}{\rightarrow}_1 p && \text{for } q, p \in Q_1 \\
q \overset{a}{\rightsquigarrow} p &:= q \overset{a}{\rightarrow}_2 p && \text{for } q, p \in Q_2 \\
q \overset{a}{\rightsquigarrow} p &:= p \in I_2 \wedge \exists p' \in F_1.\ q \overset{a}{\rightarrow}_1 p' && \text{for } q \in Q_1, p \in Q_2
\end{aligned}$$

and

$$I := \begin{cases} I_1 \cup I_2 & \text{if } \exists q \in I_1 \cap F_1 \\ I_1 & \text{otherwise} \end{cases}$$

where the constructors for the sum type $Q_1 + Q_2$ are used implicity again. Note that I is computable because the condition is decidable. The NFA first runs $\mathcal{A}_1$ and may switch to $\mathcal{A}_2$ when $\mathcal{A}_1$ could transition into a final state. The case distinction in I is necessary to handle $\varepsilon \in \mathscr{L}_R(\mathcal{A}_1)$.

$\mathscr{L}_B(\mathcal{A}_1 \cdot \mathcal{A}_2) \subseteq \mathscr{L}_R(\mathcal{A}_1) \cdot \mathscr{L}_B(\mathcal{A}_2)$: Let $\sigma \in \mathcal{A}_1 \cdot \mathcal{A}_2$ and ϱ be an accepting run. There is an n such that $\varrho[0..n)$ are states of $\mathcal{A}_1$ and $\varrho[n..]$ are states of $\mathcal{A}_2$. Then, by definition of $\rightsquigarrow$, there is a $q \in F_1$ such that $\varrho[0..n)q$ is an accepting run on $\sigma[0..n)$ for $\mathcal{A}_1$ (if $n = 0$, then $\varepsilon \in \mathscr{L}_R(\mathcal{A}_1)$ and q exists by the definition of I). Furthermore, $\varrho[n..]$ is an accepting run on $\sigma[n..]$ for $\mathcal{A}_2$ because $\rightsquigarrow$ requires $\sigma n \in I_2$ and $\mathcal{A}_1 \cdot \mathcal{A}_2$ asserts that there are infinitely many final states of $\mathcal{A}_2$ in ϱ. Hence, $\sigma[0..n) \in \mathscr{L}_R(\mathcal{A}_1)$ and $\sigma[n..] \in \mathscr{L}_B(\mathcal{A}_2)$ and so $\sigma \in \mathscr{L}_R(\mathcal{A}_1) \cdot \mathscr{L}_B(\mathcal{A}_2)$.

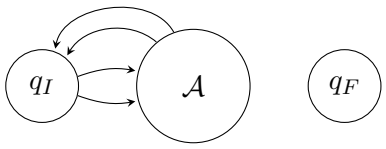


Figure 5.2: The ω-iteration a normalized NFA (compare Figure 4.1).

$\mathscr{L}_R(\mathcal{A}_1) \cdot \mathscr{L}_B(\mathcal{A}_2) \subseteq \mathscr{L}_B(\mathcal{A}_1 \cdot \mathcal{A}_2)$: Let $\sigma \in \mathscr{L}_R(\mathcal{A}_1) \cdot \mathscr{L}_B(\mathcal{A}_2)$. Then there are x and τ such that $\sigma \equiv x\tau$, $x \in \mathscr{L}_R(\mathcal{A}_1)$, and $\tau \in \mathscr{L}_B(\mathcal{A}_2)$. Let r be an accepting run on x and ϱ be an accepting run on τ. Then $r[0..|r|-1)\tau$ is an accepting run for $x\tau$ and hence $x\tau \in \mathscr{L}_B(\mathcal{A}_1 \cdot \mathcal{A}_2)$. □

Remark 5.6. *In our Coq development we prove Lemma 5.5 differently: we use $Q_1 \times Q_2$ as states. While $\mathcal{A}_1$ is running the state of $\mathcal{A}_2$ is not allowed to change and vice versa. Hence, one has a state of $\mathcal{A}_1$ and $\mathcal{A}_2$ available for all positions. Because one has to use the sum type constructors in Coq, one has to do less case distinctions. We use a normalized $\mathcal{A}_1$.*

Lemma 5.7 (ω-Iteration of NFAs)**.** *For any $\mathcal{A}_1$ one can construct an NFA $\mathcal{A}^\omega$ accepting the ω-iteration of $\mathscr{L}_R(\mathcal{A})$:*

$$\mathscr{L}_B(\mathcal{A}^\omega) \equiv \mathscr{L}_R(\mathcal{A})^\omega$$

Proof. Let $\mathcal{A} = (Q, I, F, \to)$. By Lemma 4.15 we can assume that $\mathcal{A}$ is normalized because $\mathscr{L}_R(\mathcal{A})^\omega \equiv \mathscr{L}_R^+(\mathcal{A})^\omega$. Then there are unique and distinct states $q_I \in I$ and $q_F \in F$. Let $\mathcal{A}^\omega := (Q, \{q_I\}, \{q_I\}, \rightsquigarrow)$ where

$$q \overset{a}{\rightsquigarrow} p := \begin{cases} q \overset{a}{\to} q_F & p = q_I \\ q \overset{a}{\to} p & p \neq q_I \end{cases}$$

Instead of stopping $\mathcal{A}$ in q_F, $\mathcal{A}$ is restarted by transitioning to q_I instead of q_F (see Figure 5.2 and compare Figure 4.1). This is safe because $\mathcal{A}$ is normalized.

$\mathscr{L}_B(\mathcal{A}^\omega) \subseteq \mathscr{L}_R(\mathcal{A})^\omega$: Let $\sigma \in \mathcal{A}^\omega$ and ϱ be an accepting run. To show $\sigma \in \mathscr{L}_R(\mathcal{A})^\omega$, it suffices to show by Fact 2.21 that there is a strictly monotone function f, such that $f0 = 0$ and $\sigma[fn..f(n+1)) \in \mathscr{L}_R(\mathcal{A})$ for all n. This strictly monotone function is exactly $\varrho\#I$. It it the case that $(\varrho\#I)0 = 0$ because q_I is the only initial state and $\sigma[fn..f(n+1)) \in \mathscr{L}_R(\mathcal{A})$ because no transition enters q_I in $\mathcal{A}$ and hence can only be entered in $\mathcal{A}^\omega$ when $\mathcal{A}$ would enter q_F.

$\mathscr{L}_R(\mathcal{A})^\omega \subseteq \mathscr{L}_B(\mathcal{A}^\omega)$: Let $\sigma \in \mathscr{L}_R(\mathcal{A})^\omega$. There are nonempty x_i such that $x_i \in \mathscr{L}_R(\mathcal{A})$ and $\sigma \equiv \Pi x_i$. With constructive choice for accepting finite runs by Lemma 4.14, one can construct r_is such that $q_I r_i q_F$ is an accepting run for x_i for all i (we know the first and last state of the run because $\mathcal{A}$ is normalized). Constructive choice is important because the r_is need to be

computable. Then $\Pi q_I r_i$ is final by Fact 2.15 because q_I is a final state of $\mathcal{A}^\omega$, valid on Πx_i by Lemma 4.5, and initial because q_I is an initial state. So $\sigma \in \mathscr{L}_B(\mathcal{A}^\omega)$ □

Lemma 5.8 ([2])**.** *$xy^\omega \in \mathscr{L}_B(\mathcal{A})$ is decidable.*

Proof. The languages $\{u|u = x\}$ and $\{u|u = y\}$ are regular and by Lemmas 5.5 and 5.7 we obtains an NFA $\mathcal{A}_{xy^\omega}$ with $\mathscr{L}_B(\mathcal{A}_{xy^\omega}) \equiv \{\sigma|\sigma \equiv xy^\omega\}$. We then decide emptiness of $\mathcal{A} \cap \mathcal{A}_{xy^\omega}$ using Lemma 4.21. □

Chapter 6

Complementation of Büchi Automata

Büchi automata cannot be determinized which causes complementation to be more involved. In this chapter we present the complementation construction given by Büchi [3] and give the essential correctness lemmas for ω-sequences.

For each Büchi automaton $\mathcal{A}$ there are finitely many languages L_i covering Σ^ω such that an L_i is either a subset of $\mathscr{L}_B(\mathcal{A})$ or disjoint from $\mathscr{L}_B(\mathcal{A})$. Moreover, each L_i is accepted by a Büchi automaton. The complement of $\mathcal{A}$ is the finite union of all L_i disjoint from $\mathscr{L}_B(\mathcal{A})$. The L_is are obtained as VW^ω where V and W are equivalence classes of an equivalence relation on strings of finite index. As the equivalence relation already occurs in Büchi [3], we will call it Büchi equivalence. The proof separates in two parts: *Compatibility* states that a VW^ω is either a subset of $\mathscr{L}_B(\mathcal{A})$ or disjoint from it and *totality* says that each sequence is in some VW^ω.

Totality can only be proven assuming AR. If we proved complementation only for ω-sequences, complementation would rely on AR always. To avoid this, we will factorize the totality proof in so-called admissible sequence structures in Chapter 7 and prove the complementation construction correct for AS structures. Then we can show that AR implies BC:

Definition 6.1 (BC)**.** *BC is the following proposition: Büchi automata are closed under complement and the word problem of Büchi automata is logically decidable. Formally*

$$\mathsf{BC} := \forall \mathcal{A}.\ (\exists \overline{\mathcal{A}}.\ \mathscr{L}_B(\overline{\mathcal{A}}) \equiv \overline{\mathscr{L}_B(\mathcal{A})}) \wedge (\forall \sigma.\ \sigma \in \mathscr{L}_B(\mathcal{A}) \vee \sigma \notin \mathscr{L}_B(\mathcal{A})).$$

Throughout this chapter, let $\mathcal{A}$ denote a fixed NFA which should be complemented. To construct the VW^ω languages, we need an enumerable representation of the equivalence classes of Büchi equivalence.

6.1 The Büchi Equivalence Relation

We define the Büchi equivalence relation $\sim$ using a finite type $\Sigma^*\backslash\sim$ to represent the equivalence classes of $\sim$ and a classifier $[x]_\sim$ from Σ^* to $\Sigma^*\backslash\sim$ assigning strings to its equivalence class. To define Büchi equivalence, we need two other relations:

- We say that x **transforms** q to p and write $q \overset{x}{\Longrightarrow} p$ if there is a run r such that x and r form a path from q to p and
- say that x **transforms** q to p **final** and write $q \overset{x}{\underset{F}{\Longrightarrow}} p$ if there is a run r such that x and r form a path from q to p and there is a final state in r.

Fact 6.2. *$q \overset{x}{\underset{F}{\Longrightarrow}} p$ implies $q \overset{x}{\Longrightarrow} p$.*

Fact 6.3. *$q \overset{x}{\Longrightarrow} p$ and $q \overset{x}{\underset{F}{\Longrightarrow}} p$ are decidable.*

Proof. Because $q \overset{x}{\Longrightarrow} p$ and $q \overset{x}{\underset{F}{\Longrightarrow}} p$ quantify over a finite run of length $|x|+1$, there are only finitely many runs to check. □

We want to define Büchi equivalence such that two equivalent strings agree on $q \overset{\cdot}{\Longrightarrow} p$ and $q \overset{\cdot}{\underset{F}{\Longrightarrow}} p$ for all states q and p. Because $q \overset{x}{\underset{F}{\Longrightarrow}} p$ implies $q \overset{x}{\Longrightarrow} p$, a string can either transform q to p final, transform q to p but not final, or cannot transform q to p at all. We use constants f, t, and n for the three cases. The equivalence classes are then represented as

$$\Sigma^*\backslash\sim := \{\mathsf{t},\mathsf{f},\mathsf{n}\}^{Q\times Q}.$$

The letters V, W, and U range over equivalence classes. For each string, its equivalence class is computed as:

$$([x]_\sim)_{q,p} := \begin{cases} \mathsf{f} & \text{if } q \overset{x}{\underset{F}{\Longrightarrow}} p \\ \mathsf{t} & \text{if } q \overset{x}{\Longrightarrow} p \wedge q \overset{x}{\underset{F}{\not\Longrightarrow}} p \\ \mathsf{n} & \text{if } q \overset{x}{\not\Longrightarrow} p \end{cases}$$

The string language $[\![V]\!]$ contains the strings which belong to V.

$$[\![V]\!] := \{x \mid [x]_\sim = V\}$$

A Büchi equivalence class captures the important information of a string for the Büchi automaton $\mathcal{A}$: for each pair of states q and p the equivalence class specifies whether $\mathcal{A}$ can enter p when starting in q on x and whether it can visit a final state in between. If unambiguous we write V for $[\![V]\!]$. Two

strings are **Büchi equivalent** if they are assigned to the same equivalence class.

$$x \sim y := [x]_\sim = [y]_\sim$$

By definition $\sim$ is decidable and has only finitely many equivalence classes.

Fact 6.4. *$x \sim y$ if and only if*

$$\forall qp.\ (q \xRightarrow{x} p \leftrightarrow q \xRightarrow{y} p) \wedge (q \xRightarrow[F]{x} p \leftrightarrow q \xRightarrow[F]{y} p).$$

Lemma 6.5. *$\sim$ is right congruent, i.e. if $x \sim y$ then $xu \sim yu$ for all u.*

Proof. Fact 6.4 gives $\forall qp.\ (q \xRightarrow{x} p \leftrightarrow q \xRightarrow{y} p) \wedge (q \xRightarrow[F]{x} p \leftrightarrow q \xRightarrow[F]{y} p)$. Because both strings x and y are continued with u, this implies $\forall qp.\ (q \xRightarrow{xu} p \leftrightarrow q \xRightarrow{yu} p) \wedge (q \xRightarrow[F]{xu} p \leftrightarrow q \xRightarrow[F]{yu} p)$. Then $xu \sim yu$ by Fact 6.4. □

Lemma 6.6. *For each V an NFA $\mathcal{A}_V$ accepting $[\![V]\!]$ can be constructed.*

Proof. By 4.16 it suffices to show that $[\![V]\!]$ is decidable and that $[x]_\sim$ is a classifier for $[\![V]\!]$. Decidability of $[\![V]\!]$ follows from decidability of equality on finite types.

Let $[x]_\sim = [y]_\sim$. Then $[xu]_\sim = [yu]_\sim$ by Lemma 6.5 and hence $[x]_\sim$ is right congruent.

Let $x \in [\![V]\!]$. Then $V = [x]_\sim = [y]_\sim$ and hence $y \in [\![V]\!]$. The reverse direction is symmetric and so $[x]_\sim$ refines $[\![V]\!]$.

Because $[x]_\sim$ is right congruent and refines $[\![V]\!]$, it is a classifier for $[\![V]\!]$. □

Remark 6.7. *Unfortunately, we do not have the translation from classifiers to NFAs available in our development for technical reasons. We formalized the proof of Lemma 6.6 assuming the translation. Additionally we construct $\mathcal{A}_V$ from $\mathcal{A}$ to obtain a closed formal development. This construction is not as elegant as the proof using classifiers in Lemma 6.6.*

6.2 Compatibility of the Büchi Equivalence Classes

We are going to prove that the languages VW^ω are compatible with $\mathscr{L}_B(\mathcal{A})$: either VW^ω is a subset of $\mathscr{L}_B(\mathcal{A})$ or disjoint from it. Proving compatibility requires the three following facts for constructing finite runs.

Fact 6.8. *If $x \sim y$ and $q \cdot r \cdot p$ is valid on x, then we can construct s such that $q \cdot s \cdot p$ is valid on y.*

Proof. Such a run exists because $x \sim y$. It can be constructed because the length of s must be $|y| - 1$. □

Fact 6.9. *If $x \sim y$, $q \cdot r \cdot p$ is valid on x, and $q \cdot r \cdot p$ contains a final state, then we can construct s such that $q \cdot s \cdot p$ is valid on y and contains a final state.*

Proof. Analogous as for Fact 6.8. □

Fact 6.10. *If $\varepsilon \sim y$, then for all p we can construct s such that if $y \neq \varepsilon$, then $p \cdot s \cdot p$ is valid on y. Otherwise p is valid on y trivially.*

Proof. Note that $p \overset{\varepsilon}{\Longrightarrow} p$ for all p. The proof is analogous as for Fact 6.8 with a case distinction whether $y = \varepsilon$. □

Lemma 6.11 (Compatibility)**.** *If $\sigma \in VW^\omega \cap \mathscr{L}_B(\mathcal{A})$ for some σ, then $VW^\omega \subseteq \mathscr{L}_B(\mathcal{A})$.*

Proof. Let $\tau \in VW^\omega$. We need to show $\tau \in \mathscr{L}_B(\mathcal{A})$. Because σ and τ are in VW^ω there are $x \in V$, $u \in V$, and nonempty $y_i \in W$ and $v_i \in W$ for all i with $\sigma \equiv x \cdot \Pi y_i$ and $\tau \equiv u \cdot \Pi v_i$. Let ϱ be an accepting run for $x \cdot \Pi y_i$. Then we can cut ϱ according to $x \cdot \Pi y_i$: $\varrho \equiv r \cdot \Pi q_i s_i$. Because the y_i are nonempty, the part of ϱ on y_i is nonempty as well and q_i always exists. The following is true:

a) rq_0 is valid on x,

b) $q_i s_i q_{i+1}$ is valid on y_i for all i (note that we need to take q_{i+1} because the run needs to be of length $|x_i| + 1$),

c) $(rq_0)[0] \in I$ because ϱ is initial, and

d) because ϱ is final, a $q_i s_i$ contains a final state infinitely often by Fact 2.15.

We want to give an accepting run $r' \cdot \Pi q_i s_i'$ for $u \cdot \Pi v_i$. If $x = \varepsilon$, then r' is obtained from r by Fact 6.10 and otherwise by Fact 6.8. This guarantees that $(r'q_0)[0] = (rq_0)[0]$ and that $r'q_0$ is valid on u.

If $q_i s_i$ contains a final state, s_i' is obtained from s_i by Fact 6.9 and otherwise by Fact 6.8. This guarantees that $q_i s_i' q_{i+1}$ is valid on v_i and that $q_i s_i'$ contains a final state if $q_i s_i$ does. Figure 6.1 illustrates the construction.

The run $r' \cdot \Pi q_i s_i'$ is valid on $u \cdot \Pi v_i$ by Fact 4.2 and Lemma 4.5. It is initial because $(r'q_0)[0] = (rq_0)[0] \in I$. Lastly, the run is final by Fact 2.15 because there are infinitely many $q_i s_i'$ containing final states. □

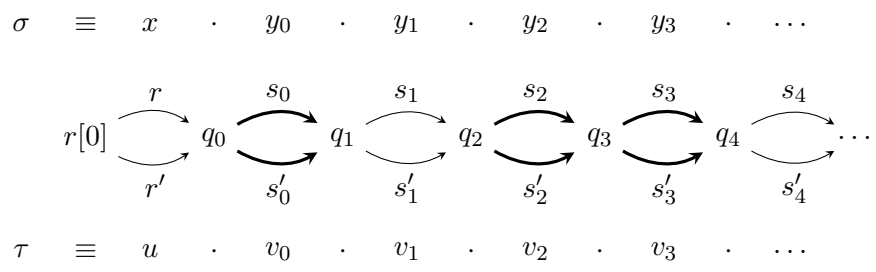


Figure 6.1: The accepting run of τ constructed from the accepting run of σ in the proof of Lemma 6.11. Bold arrows indicate runs that contain final states. This figure assumes that $u \neq \varepsilon$.

6.3 Totality of the Büchi Equivalence Classes

We want to show that for any σ there are V and W such that $\sigma \in VW^\omega$, which means that the VW^ω cover Σ^ω. We call this property *totality*. Totality cannot be proven constructively[1]. One usually proves totality using Ramsey's Theorem but AR suffices. In Chapter 9, we will see that AR is a necessary assumption to show BC.

To prove totality, we want to color strings with their Büchi equivalence class. To use AR, an operation $V+W$ yielding a finite semigroup $(\Sigma^*\!\setminus\sim, +)$ is required. The operation $V + W$ should fulfill the equation $[x]_\sim + [y]_\sim = [xy]_\sim$.

$$(V+W)_{q,p} := \begin{cases} \mathsf{f} & \text{if } \exists q'.\ (V_{q,q'} = \mathsf{f} \wedge W_{q',p} \neq \mathsf{n}) \vee (V_{q,q'} \neq \mathsf{n} \wedge W_{q',p} = \mathsf{f}) \\ \mathsf{t} & \text{otherwise if } \exists q'.\ V_{q,q'} = W_{q',p} = \mathsf{t} \\ \mathsf{n} & \text{otherwise} \end{cases}$$

In the first case, a string $xy \in V + W$ transforms q to p final if there is an indermediate state q' such that x transforms q to q', y transforms q' to p, and one of x and y does it final. Then there is a final state between q and p. The second case requires that there is an intermediate state q' to which both strings only transform but not final. In the last case no intermediate state can be found.

Proving the following two lemmas formally is tedious, because of packing and unpacking of vectors and quantifiers in the definitions of $V + W$. We only give the proof ideas and refer to our formal development for full details.

Lemma 6.12. $[xy]_\sim = [x]_\sim + [y]_\sim$.

Proof. By the definition of $V + W$ (where we ensure the desired behavior as described before). □

[1]Totality is equivalent to AR. We show that AR implies totality and see later that totality implies BC and that BC implies RF (equivalent to AR), which is independent.

Lemma 6.13. $(\Sigma^*\setminus\sim, +)$ *is a finite semigroup.*

Proof. We need to show that $+$ is associative: $V+(W+U) = (V+W)+U$. This holds because the order in which one searches for the intermediate states does not matter. There is a state q' between V and $W+U$ and a state p' between W and U if and only if p' is between $V+W$ and U and q' is between V and W. □

Lemma 6.14 (Totality)**.** *Given* AR*, for every* σ *there are* V *and* W *such that* $\sigma \in VW^\omega$*.*

Proof. By Lemma 6.12, the coloring $fnm := [\sigma[n..m)]_\sim$ is additive in the finite semigroup $(\Sigma^*\setminus\sim, +)$. By AR, there is a strictly monotone and monochrome g: $W := [\sigma[g0..g1)]_\sim$ and $[\sigma[gn..gm)]_\sim = W$ for all $n < m$.

Let $V := [\sigma[0..g0)]_\sim$. We need to show $\sigma \in VW^\omega$. By construction $\sigma[0..g0) \in V$. To show $\sigma[g0..] \in W^\omega$, with Fact 2.21 it suffices to give a strictly monotone function h such that $h0 = 0$ and $\sigma[g0..][hn..h(n+1)) \in W$. Let $hn := gn - g0$ and $h0 = 0$ follows. Hence, $\sigma[g0..][hn..h(n+1)) = \sigma[gn..g(n+1)) \in W$. □

We define the NFA accepting the complement of $\mathscr{L}_B(\mathcal{A})$ and argue shortly that it is correct for given AR. The correctness proof is given in the next chapter.

Definition 6.15 (Complement NFA)**.**

$$\mathcal{A}^C := \bigcup_{\{V,W \mid VW^\omega \cap \mathscr{L}_B(\mathcal{A}) = \emptyset\}} \mathcal{A}_V {\mathcal{A}_W}^\omega$$

By totality $\bigcup_{V,W} VW^\omega \equiv \Sigma^\omega$ and by compatibility a VW^ω is either contained in $\mathscr{L}_B(\mathcal{A})$ or in $\overline{\mathscr{L}_B(\mathcal{A})}$. As there are only finitely many VW^ω, it suffices to decide $VW^\omega \subseteq \mathscr{L}_B(\mathcal{A})$ to construct the NFA for the union. This follows from decidability of language emptiness of $\mathcal{A}_V {\mathcal{A}_W}^\omega \cap \mathcal{A}$.

Chapter 7

Admissible Sequence Structures

The previous chapter contains the crucial proofs for closure under complement of Büchi automata using AR. Because translating S1S formulas to Büchi automata requires complementation, all results would depend on AR. To develop a completely constructive view, we need to restrict the infinity of ω-sequences. We use UP sequences for that purpose. We abstract from the two kinds of sequences, ω-sequences and UP sequences, to admissible sequence structures, or short AS structures. Then we do most of the following proofs for AS structures and can use the results for UP sequences and under AR for ω-sequences.

A **sequence structure** is a structure containing a type constructor $\mathscr{A}$ and operations $\mathsf{C}_{\mathscr{A}}, \otimes_{\mathscr{A}}, \circ_{\mathscr{A}}$, and $@_{\mathscr{A}}$ of the following types:

- $\mathscr{A}$: finite Type $\to$ Type
- $\mathsf{C}_{\mathscr{A}} : \forall\Sigma.\ \mathscr{A}(\Sigma) \to \Sigma^\omega$
- $\circ_{\mathscr{A}} : \forall\Sigma\Gamma.\ \mathscr{A}(\Sigma) \to (\Sigma \to \Gamma) \to \mathscr{A}(\Gamma)$
- $\otimes_{\mathscr{A}} : \forall\Sigma\Gamma.\ \mathscr{A}(\Sigma) \to \mathscr{A}(\Gamma) \to \mathscr{A}(\Sigma \times \Gamma)$
- $@_{\mathscr{A}} : \forall\Sigma.\ \Sigma \to \mathbb{N} \to \Sigma \to \mathscr{A}(\Sigma)$

These operations must be compatible with the operations $\otimes$, $\circ$, and $@$ for ω-sequences given in Chapter 2:

- $\mathsf{C}_{\mathscr{A}}(\sigma \circ_{\mathscr{A}} f) \equiv \mathsf{C}_{\mathscr{A}}\sigma \circ f$
- $\mathsf{C}_{\mathscr{A}}(\sigma \otimes_{\mathscr{A}} \tau) \equiv \mathsf{C}_{\mathscr{A}}\sigma \otimes \mathsf{C}_{\mathscr{A}}\tau$
- $\mathsf{C}_{\mathscr{A}}(a@^n_{\mathscr{A}}b) \equiv a@^n b$

We call such a sequence structure $\mathscr{A}$**-structure**. Elements $\sigma \in \mathscr{A}(\Sigma)$ are called $\mathscr{A}$**-sequences** or only sequences if it is clear from the context that we refer to $\mathscr{A}$-sequences and not to ω-sequences. The prior compatibilities are needed to lift results from ω-sequences to $\mathscr{A}$-sequences. We use the operation $\mathsf{C}_{\mathscr{A}}$ converting $\mathscr{A}$-sequences to ω-sequences implicitly and hence for $\sigma, \tau \in \mathscr{A}(\Sigma)$ we can write σn, $\sigma \equiv \tau$, or $\sigma \in \mathscr{L}_B(\mathcal{A})$. We will leave out the subscripts of the operations if it is clear that and to which sequence structure we refer. When working with sequence structures, we need to convert ω-languages to $\mathscr{A}$-languages. For $L \subseteq \Sigma^\omega$ we write $\mathscr{A}(L)$ for the $\mathscr{A}$-language obtained from L:

$$\mathscr{A}(L) := \{\sigma \in \mathscr{A}(\Sigma) \mid \mathsf{C}_{\mathscr{A}}\sigma \in L\}$$

We need to generalize the definition of the image and preimage from ω-sequences to $\mathscr{A}$-sequences. Given a language $L \subseteq \mathscr{A}(\Sigma)$, a function f from Σ to Γ, and a function g from Γ to Σ, we define the image and preimage of L analogous to ω-languages in Chapter 2:

$$\begin{aligned} f(L) &:= \{\sigma \in \mathscr{A}(\Gamma) | \exists \tau \in L.\ \sigma \equiv \tau \circ_{\mathscr{A}} f\} \\ g^{-1}(L) &:= \{\sigma \in \mathscr{A}(\Gamma) | \sigma \circ_{\mathscr{A}} g \in L\} \end{aligned}$$

Languages of Büchi automata get restricted to the sequence structure:

$$\mathscr{L}_{\mathscr{A}}(\mathcal{A}) := \mathscr{A}(\mathscr{L}_B(\mathcal{A}))$$

An **admissible sequence structure** $\mathscr{A}$, or short AS structure, is a sequence structure satisfying the following additional properties regarding NFAs: For any NFA $\mathcal{A}$

a) it is decidable whether $\mathscr{L}_{\mathscr{A}}(\mathcal{A}) \equiv \emptyset$ or $\exists \sigma.\ \sigma \in \mathscr{L}_{\mathscr{A}}(\mathcal{A})$,

b) the image construction is correct, meaning that $\mathscr{L}_{\mathscr{A}}(f(\mathcal{A})) \equiv f(\mathscr{L}_{\mathscr{A}}(\mathcal{A}))$, and

c) totality holds: for all $\sigma \in \mathscr{A}(\Sigma)$ there are V and W such that $\sigma \in VW^\omega$.

Condition a) is needed to show decidability of satisfiability of S1S. As the image construction for Büchi automata will be needed later and cannot be proven correct for sequence structures in general, b) is required. Last, c) is needed to prove the complement construction correct for AS structures.

AS structures are needed to prove correctness, not to build operations. They only provide proofs. All following constructions will result in the same Büchi automata independent of the AS structure.

7.1 Correctness of Complementation

Correctness of operations on Büchi automata apart of image and complementation with respect to an AS structure follows from correctness on ω-sequences. Complementation can be proven correct because AS structures provide a proof of totality. Let $\mathscr{A}$ be an AS structure and $\mathcal{A}$, $\mathcal{A}_1$, and $\mathcal{A}_2$ be NFAs over Σ.

Lemma 7.1. *The union, intersection, and preimage construction for Büchi automata are correct for $\mathscr{A}$-sequences:*

$$\begin{aligned}
\mathscr{L}_{\mathscr{A}}(\mathcal{A}_1 \cup \mathcal{A}_2) &\equiv \mathscr{L}_{\mathscr{A}}(\mathcal{A}_1) \cup \mathscr{L}_{\mathscr{A}}(\mathcal{A}_2) \\
\mathscr{L}_{\mathscr{A}}(\mathcal{A}_1 \cap \mathcal{A}_2) &\equiv \mathscr{L}_{\mathscr{A}}(\mathcal{A}_1) \cap \mathscr{L}_{\mathscr{A}}(\mathcal{A}_2) \\
\mathscr{L}_{\mathscr{A}}(f^{-1}(\mathcal{A})) &\equiv f^{-1}(\mathscr{L}_{\mathscr{A}}(\mathcal{A})) \qquad \text{for } f \text{ from } \Sigma \text{ to } \Gamma
\end{aligned}$$

Proof. Follows immediately from Lemmas 5.3, 5.4, and 5.1. □

We prove correctness of the complement construction with respect to $\mathscr{A}$-sequences.

Lemma 7.2. $\mathscr{L}_{\mathscr{A}}(\mathcal{A}) \cap \mathscr{L}_{\mathscr{A}}(\mathcal{A}^C) \equiv \emptyset$.

Proof. Assume that there is a $\sigma \in \mathscr{L}_{\mathscr{A}}(\mathcal{A}) \cap \mathscr{L}_{\mathscr{A}}(\mathcal{A}^C)$. By definition of $\mathcal{A}^C$ there are V and W such that $\sigma \in VW^\omega$ and $VW^\omega \cap \mathscr{L}_B(\mathcal{A}) \equiv \emptyset$. This is a contradiction since $\sigma \in VW^\omega$ and $\sigma \in \mathscr{L}_{\mathscr{A}}(\mathcal{A})$ implying $\sigma \in \mathscr{L}_B(\mathcal{A})$. □

Lemma 7.3. *If $VW^\omega \cap \mathscr{L}_B(\mathcal{A}) \equiv \emptyset$, then $\mathscr{A}(VW^\omega) \subseteq \mathscr{L}_{\mathscr{A}}(\mathcal{A}^C)$.*

Proof. By definition of $\mathcal{A}^C$. □

Lemma 7.4. *If $VW^\omega \cap \mathscr{L}_B(\mathcal{A}) \not\equiv \emptyset$, then $\mathscr{A}(VW^\omega) \subseteq \mathscr{L}_{\mathscr{A}}(\mathcal{A})$.*

Proof. Let $\sigma \in \mathscr{A}(VW^\omega)$. By Lemma 4.21 there is a $\tau \in VW^\omega \cap \mathscr{L}_B(\mathcal{A})$ because $VW^\omega \cap \mathscr{L}_B(\mathcal{A}) \not\equiv \emptyset$. Since $\sigma \in \mathscr{A}(\Sigma)$ it suffices to show that $\sigma \in \mathscr{L}_B(\mathcal{A})$. This follows by compatibility (Lemma 6.11). □

Lemma 7.5. $\mathscr{L}_{\mathscr{A}}(\mathcal{A}) \cup \mathscr{L}_{\mathscr{A}}(\mathcal{A}^C) \equiv \mathscr{A}(\Sigma)$

Proof. $\mathscr{L}_{\mathscr{A}}(\mathcal{A}) \cup \mathscr{L}_{\mathscr{A}}(\mathcal{A}^C) \subseteq \mathscr{A}(\Sigma)$ follows directly. Let $\sigma \in \mathscr{A}(\Sigma)$. By the totality proof of $\mathscr{A}$ there are V and W such that $\sigma \in VW^\omega$. If $VW^\omega \cap \mathscr{L}_B(\mathcal{A}) \equiv \emptyset$ (which is decidable) $\sigma \in \mathscr{L}_{\mathscr{A}}(\mathcal{A}^C)$ by Lemma 7.3. Otherwise $\sigma \in \mathscr{L}_{\mathscr{A}}(\mathcal{A})$ by Lemma 7.4. □

Theorem 7.6 (Closure under Complement)**.** $\mathscr{L}_{\mathscr{A}}(\mathcal{A}^C) \equiv \overline{\mathscr{L}_{\mathscr{A}}(\mathcal{A})}$.

Proof. By Lemmas 7.2 and 7.5. □

Another consequence of totality and the complement construction is logical decidability of the word problem for $\mathscr{A}$-sequences.

Corollary 7.7. *For any $\sigma \in \mathscr{A}(\Sigma)$, $\sigma \in \mathscr{L}_{\mathscr{A}}(\mathcal{A})$ is logically decidable.*

Proof. By Lemma 7.5. □

The full logical strength of closure under complementation is captured only in Theorem 7.6 and Corollary 7.7 together as we need both to show that AR implies BC.

We present two AS structures, the ω-structure and the UP-structure.

7.2 The ω-Structure

The structure of all ω-sequences is called the ω-structure: $\omega(\Sigma) := \Sigma^\omega$ with C_ω as the identity. The ω-structure is only admissible given AR because then by Lemma 6.14 there is a proof of totality.

Corollary 7.8. *AR implies BC.*

Proof. Let $\mathcal{A}$ be an NFA. Given AR the ω-structure is admissible and $\mathcal{A}^C$ accepts the complement language of $\mathcal{A}$ by Theorem 7.6. The word problem is logically decidable by Corollary 7.7. □

Note that BC is weaker than the complement construction. BC only requires NFAs accepting the complement to exists, but not their construction.

7.3 The UP-Structure

We call the structure representing UP sequences the UP-structure. UP sequences are represented by pairs of strings: $\mathsf{UP}(\Sigma) := \Sigma^* \times \Sigma^+$ and $\mathsf{C}_{\mathsf{UP}}(x, y) := xy^\omega$. We already proved that UP sequences are closed under $\circ$ in Fact 2.22, under $\otimes$ in Lemma 2.25, and under @ in Fact 2.26. The construction in these proofs can directly be used to define operations $\circ_{\mathsf{UP}}$, $\otimes_{\mathsf{UP}}$, and $@_{\mathsf{UP}}$. Language emptiness is decidable by Lemma 4.21. Totality for UP sequences becomes trivial.

Fact 7.9. *For every xy^ω there are V and W such that $xy^\omega \in VW^\omega$.*

Proof. Choose $V := [x]_\sim$ and $W := [y]_\sim$. □

To show that the UP-structure is admissible, it remains to show that the image construction for Büchi automata from Lemma 5.2 is correct. This is more effort for UP sequences as for ω-sequences, because the strings representing the UP sequences may be of different length.

Lemma 7.10. *If $xy^\omega \in \mathscr{L}_B(\mathcal{A})$, then there are r and s such that rs^ω is an accepting run for xy^ω.*

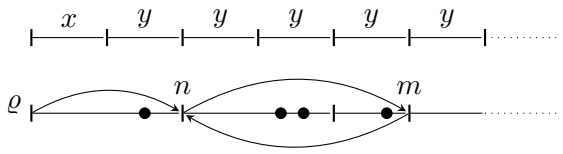


Figure 7.1: The UP run constructed in Lemma 7.10. Dots indicate final states in ϱ and vertical lines potentially loop positions.

Proof. Let ϱ be an accepting run. We say that i is a **potentially loop position** if there is a final state in the part of ϱ on the i-th repetition of y, or formally $\exists n.\ |x| + i \cdot |y| < n \leq |x| + (i+1) \cdot |y| \wedge \varrho n \in F$. Clearly, being a potentially loop position is decidable and there are infinitely many potentially loop positions because there are infinitely many final states in ϱ. Thus, there are two potentially loop positions $i < j$ with $\varrho n = \varrho m$ where $n = |x| + (i+1) \cdot |y|$ and $m = |x| + (j+1) \cdot |y|$. Because n and m are aligned with the repetitions of y, $\varrho[0..n)\varrho[n..m)^\omega$ is a valid and initial run on xy^ω. The run is final because $n \leq |x| + j \cdot |y|$ and there is a final state between $|x| + j \cdot |y|$ and m (because j is a potentially loop position). Figure 7.1 shows the UP run. □

Lemma 7.11. *If $xy^\omega \in \mathscr{L}_B(\mathcal{A})$, then there are x', y', r' and s' such that $x'y'^\omega \equiv xy^\omega$, $|x'| = |r'|$, $|y'| = |s'|$, and $r's'^\omega$ is an accepting run for $x'y'^\omega$.*

Proof. Let rs^ω be an accepting run for $xy^\omega \in \mathscr{L}_B(\mathcal{A})$ by Lemma 7.10. By Lemma 2.25 there are u and v with $xy^\omega \otimes rs^\omega \equiv uv^\omega$. Then $x' := \pi_1 u$, $y' := \pi_1 v$, $r' := \pi_2 u$, and $s' := \pi_2 v$ complete the proof. □

Lemma 7.12. $\mathscr{L}_{\mathsf{UP}}(f(\mathcal{A})) \equiv f(\mathscr{L}_{\mathsf{UP}}(\mathcal{A}))$

Proof. $\mathscr{L}_{\mathsf{UP}}(f(\mathcal{A})) \supseteq f(\mathscr{L}_{\mathsf{UP}}(\mathcal{A}))$ follows by correctness of $f(\mathcal{A})$ for ω-sequences (Lemma 5.2).

Let $xy^\omega \in \mathscr{L}_{\mathsf{UP}}(f(\mathcal{A}))$ and rs^ω be an accepting run with $|x| = |r|$ and $|y| = |s|$ by Lemma 7.11. By the definition of the transition function, there is an $a \in \Sigma$ such that $(rs^\omega)n \xrightarrow{a}_{\mathcal{A}} (rs^\omega)(n+1)$ and $fa = (xy^\omega)n$ for all n. With constructive choice one obtain an $a(n)$. Let $u := a(0) \dots a(|x| - 1)$ and $v := a(|x|) \dots a(|x| + |y| - 1)$. Because $|u| = |r|$ and $|v| = |s|$, rs^ω is an accepting run for uv^ω for $\mathcal{A}$ and $xy^\omega \in f(\mathscr{L}_{\mathsf{UP}}(\mathcal{A}))$. □

7.4 BC is equivalent to BU

The UP-structure is admissible without assumptions and thus the complement construction from Chapter 6 is correct for UP sequences constructively. This yields a different and equivalent characterization of BC:

Definition 7.13 (BU)**.** *BU is the following proposition: If the languages of two Büchi automata contain the same UP sequences, then the languages are equivalent. Formally*

$$\mathsf{BU} := \forall \mathcal{A}_1 \mathcal{A}_2.\ \mathscr{L}_{\mathsf{UP}}(\mathcal{A}_1) \equiv \mathscr{L}_{\mathsf{UP}}(\mathcal{A}_2) \to \mathscr{L}_B(\mathcal{A}_1) \equiv \mathscr{L}_B(\mathcal{A}_2).$$

Interestingly, BU does not require logical decidability of the word problem or something else. It expresses the idea that two Büchi automata, which are finite transition systems and which accept different ω-languages, accept different languages of finitely representable UP sequences. We begin to show that BC implies BU.

Lemma 7.14. *BC implies BU.*

Proof. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two NFAs with $\mathscr{L}_{\mathsf{UP}}(\mathcal{A}_1) \equiv \mathscr{L}_{\mathsf{UP}}(\mathcal{A}_2)$. We show that $\mathscr{L}_B(\mathcal{A}_1) \subseteq \mathscr{L}_B(\mathcal{A}_2)$. By symmetry this implies $\mathscr{L}_B(\mathcal{A}_1) \equiv \mathscr{L}_B(\mathcal{A}_2)$.

By BC there is an $\overline{\mathcal{A}_2}$ with $\overline{\mathscr{L}_B(\mathcal{A}_2)} \equiv \mathscr{L}_B(\overline{\mathcal{A}_2})$. By Lemma 4.21 we make a case analysis on the emptiness of $\mathcal{A}_1 \cap \overline{\mathcal{A}_2}$.

If $\mathscr{L}_B(\mathcal{A}_1 \cap \overline{\mathcal{A}_2})$ is empty, let $\sigma \in \mathscr{L}_B(\mathcal{A}_1)$ and we need to show $\sigma \in \mathscr{L}_B(\mathcal{A}_2)$. Using BC we have $\sigma \in \mathscr{L}_B(\mathcal{A}_2) \vee \sigma \notin \mathscr{L}_B(\mathcal{A}_2)$. If $\sigma \notin \mathscr{L}_B(\mathcal{A}_2)$ then $\sigma \in \mathscr{L}_B(\overline{\mathcal{A}_2})$ and $\sigma \in \mathscr{L}_B(\mathcal{A}_1 \cap \overline{\mathcal{A}_2})$ contradicting emptiness.

Otherwise, there is an $xy^\omega \in \mathscr{L}_B(\mathcal{A}_1 \cap \overline{\mathcal{A}_2})$ by Corollary 4.22 and we have $xy^\omega \in \mathscr{L}_B(\mathcal{A}_1)$ and $xy^\omega \notin \mathscr{L}_B(\mathcal{A}_2)$, which contradicts that $\mathcal{A}_1$ and $\mathcal{A}_2$ accept the same UP sequences. □

To show that BU implies BC we first show that BU and Lemmas 7.2 and 7.5 instantiated for the UP-structure imply the instantiation of the same lemmas for the ω-structure.

Lemma 7.15. *BU implies for any NFA $\mathcal{A}$ that $\mathscr{L}_B(\mathcal{A}) \cup \mathscr{L}_B(\mathcal{A}^C) \equiv \Sigma^\omega$.*

Proof. Surely there is an NFA $\mathcal{A}_{univ}$ accepting Σ^ω (e.g. one final and initial state always transitioning into itself). Then the claim is equivalent to $\mathscr{L}_B(\mathcal{A} \cup \mathcal{A}^C) \equiv \mathscr{L}_B(\mathcal{A}_{\mathsf{univ}})$. With BU it suffices to show $\mathscr{L}_{\mathsf{UP}}(\mathcal{A} \cup \mathcal{A}^C) \equiv \mathscr{L}_{\mathsf{UP}}(\mathcal{A}_{\mathsf{univ}})$, which follows by Lemma 7.5 instantiated for the UP-structure. □

Lemma 7.16. *BU implies for any NFA $\mathcal{A}$ that $\mathscr{L}_B(\mathcal{A}) \cap \mathscr{L}_B(\mathcal{A}^C) \equiv \emptyset$.*

Proof. There is an NFA $\mathcal{A}_\emptyset$ accepting $\emptyset$ (e.g. no states at all). The claim is equivalent to $\mathscr{L}_B(\mathcal{A} \cap \mathcal{A}^C) \equiv \mathscr{L}_B(\mathcal{A}_\emptyset)$. With BU it suffices to show $\mathscr{L}_{\mathsf{UP}}(\mathcal{A} \cap \mathcal{A}^C) \equiv \mathscr{L}_{\mathsf{UP}}(\mathcal{A}_\emptyset) \equiv \emptyset$, which follows from Lemma 7.2 for the UP-structure. □

Lemma 7.17. *BU implies BC.*

Proof. Given BU, the word problem of Büchi automata is logically decidable by Lemma 7.15. Büchi automata are closed under complement because for any $\mathcal{A}$ the NFA $\mathcal{A}^C$ accepts $\overline{\mathscr{L}_B(\mathcal{A})}$ by Lemmas 7.15 and 7.16. □

Theorem 7.18. *BU is equivalent to BC.*

Proof. By Lemmas 7.14 and 7.17. □

Chapter 8

Monadic Second Order Logic S1S

In this chapter, we show that satisfiability in S1S is decidable and that satisfaction is logically decidable. Following for instance Thomas [22] and Blumensath [1], we first look at a simplified core logic MSO_0 that only uses second order variables. MSO_0 formulas are translated to Büchi automata.

Decidability of language emptiness yields decidability of satisfiability. Analogously, logical decidability of the word problem yields logical decidability of satisfaction. We reduce the logic with both first and second order variables, called MSO, to MSO_0. Classically, second order variables are interpreted with sets of numbers. Since in type theory there are no sets, we interpret second order variables as $\mathscr{A}$-sequences over $\mathbb{B}$. This makes sense as we need to construct Büchi automata accepting them. With this choice we restrict S1S to decidable sets.

We define S1S for arbitrary AS structures because we can instantiate the results constructively in the UP-structure or under AR in the ω-structure. Let $\mathscr{A}$ by an arbitrary AS structure throughout this chapter.

8.1 The Core Logic MSO_0

MSO_0 **formulas** are given by the grammar

$$\varphi, \psi ::= X \lessdot Y \mid X \subseteq Y \mid \varphi \wedge \psi \mid \neg\varphi \mid \exists X.\ \varphi$$

where $X, Y \in \mathbb{N}$. An **interpretation** is a function from $\mathbb{N}$ to $\mathscr{A}(\mathbb{B})$ assigning an $\mathscr{A}$-sequence over $\mathbb{B}$ to each variable. Satisfaction in MSO_0 is defined by

$$
\begin{aligned}
J \vDash_0 X \lessdot Y &:= \exists n \in (JX).\ \exists m \in (JY).\ m < n \\
J \vDash_0 X \subseteq Y &:= \forall n.\ n \in (JX) \to n \in (JY) \\
J \vDash_0 \varphi \wedge \psi &:= J \vDash_0 \varphi \wedge J \vDash_0 \psi \\
J \vDash_0 \neg\varphi &:= J \nvDash_0 \varphi \\
J \vDash_0 \exists X.\ \varphi &:= \exists M.\ J_{X:=M} \vDash_0 \varphi
\end{aligned}
$$

where $n \in N := N\ n = \mathsf{true}$. $J_{X:=M}$ is the interpretation assigning X to M and all other variables according to J. Formally:

$$J_{X:=M} := \lambda Y.\mathsf{if}\ X = Y\ \mathsf{then}\ M\ \mathsf{else}\ JY$$

We speak of sequences over $\mathbb{B}$ as sets for convenience and use letters M and N for sets. Note that the satisfaction relation uses sequences only as sets. Because we use sequences, we need to use sequence equivalence instead of equality. The empty set $\emptyset$ can be obtained by $\mathsf{false}@^0\mathsf{false}$ always (cf. Chapter 2 for the @ operation).

We define a function $\mathcal{V}\varphi$ yielding a list of the **free variables** of φ:

$$
\begin{aligned}
\mathcal{V}(X \lessdot Y) &:= [X; Y] \\
\mathcal{V}(X \subseteq Y) &:= [X; Y] \\
\mathcal{V}(\varphi \wedge \psi) &:= \mathcal{V}\varphi \# \mathcal{V}\psi \\
\mathcal{V}(\neg\varphi) &:= \mathcal{V}\varphi \\
\mathcal{V}(\exists X.\ \varphi) &:= \mathcal{V}\varphi \setminus X
\end{aligned}
$$

We say that two interpretations J and J' **agree** on a list of variables l if $JX \equiv J'X$ for all variables X contained in l.

Lemma 8.1 (Coincidence)**.** *If J and J' agree on the free variables of φ, then $J \vDash_0 \varphi$ if and only if $J' \vDash_0 \varphi$.*

Proof. By induction on φ. □

8.2 Translation of Formulas to Büchi Automata

The coincidence Lemma 8.1 shows that only the sets assigned to free variables matter. We use this to define the **vector language** of a formula φ. This language contains the interpretations satisfying φ reduced to the free variables of φ. Because there are only finitely many free variables, we only need to consider finitely many sets. Hence, we can take the product of them to obtain a sequence over $\mathbb{B}^{\mathcal{V}\varphi}$. Such a sequence is called a **vector interpretation**.

A vector interpretation $\sigma \in \mathscr{A}\left(\mathbb{B}^{\mathcal{V}\varphi}\right)$ satisfies φ, written $\sigma \vDash_0 \varphi$, if

$$\lambda X.\mathsf{if}\ X \in \mathcal{V}\varphi\ \mathsf{then}\ \pi_X\sigma\ \mathsf{else}\ \emptyset$$

satisfies φ. The choice of the empty set for non free variables is arbitrary.

Definition 8.2 (Vector Language)**.** *The vector language of a formula φ is the language of vector interpretations satisfying φ:*

$$\mathscr{L}_V(\varphi) := \left\{\sigma \in \mathscr{A}\left(\mathbb{B}^{\mathcal{V}\varphi}\right) \,\middle|\, \sigma \vDash_0 \varphi\right\}$$

Theorem 8.3. *For all formulas φ, an interpretation J satisfies φ if and only if the vector interpretation $\bigotimes_{\mathcal{V}\varphi} J$ satisfies φ.*

Proof. By coincidence (Lemma 8.1). □

It suffices to look at vector interpretations by the previous theorem. As sets are $\mathscr{A}$-sequences by definition, $\bigotimes_{\mathcal{V}\varphi} J$ is an $\mathscr{A}$-sequence, too. To show decidability of satisfiability of MSO_0 formulas, we translate a formula into a Büchi automaton accepting the vector language of the formula. Before constructing the Büchi automaton, we need to introduce some notations for vectors. Let l, l_1 and l_2 be lists of variables.

- We write $\mathbb{B}^l$ for the finite type $\mathbb{B}^{\Sigma X. X \in l}$ (where $X \in l$ is assumed to be proof irrelevant).
- If l_2 contains only variables of l_1, we write $\Downarrow_{l_2}^{l_1}$ for the function projecting vectors $\mathbb{B}^{l_1}$ to $\mathbb{B}^{l_2}$.
- If X is contained in l, we silently identify $\mathbb{B}^{l\setminus X} \times \mathbb{B}$ with $\mathbb{B}^l$.
- If l_1 contains exactly the same variables as l_2, we identify $\mathbb{B}^{l_1}$ with $\mathbb{B}^{l_2}$.

Let $L \subseteq \mathscr{A}(\mathbb{B}^l)$. The **outprojection** of X from L is the following:

$$L \downarrow X := \left\{\sigma \in \mathscr{A}\left(\mathbb{B}^{l\setminus X}\right) \,\middle|\, \begin{array}{ll} \exists \tau \in \mathscr{A}(\mathbb{B}).\ \sigma \otimes \tau \in L & \text{if } X \text{ is in } l \\ \sigma \in L & \text{otherwise} \end{array}\right\}$$

If X is not in L, L and $L \downarrow X$ are equivalent modulo type identification. Otherwise $L \downarrow X$ is the language of sequences over $\mathbb{B}^{l\setminus X}$ that can be extended to sequences in L.

Lemma 8.4. *If $L \subseteq \mathscr{A}(\mathbb{B}^l)$ is extensional, then $L \downarrow X \equiv \Downarrow_{l\setminus X}^{l}(L)$.*

Proof. If X is not in l, both $L \downarrow X$ and $\Downarrow_{l\setminus X}^{l}(L)$ are equivalent to L using extensionality of L. If X is in l, $L \downarrow X$ projects outs a sequence over $\mathbb{B}$ at once, where $\Downarrow_{l\setminus X}^{l}(L)$ projects out a boolean at each position (which then form a sequence). This is equivalent by extensionality of L. □

Lemma 8.5. *The following equivalences for vector languages hold:*

$$\begin{aligned} \mathscr{L}_V(\neg\varphi) &\equiv \overline{\mathscr{L}_V(\varphi)} \\ \mathscr{L}_V(\varphi \wedge \psi) &\equiv \left(\Downarrow_{\mathcal{V}\varphi}^{\mathcal{V}(\varphi\wedge\psi)}\right)^{-1}(\mathscr{L}_V(\varphi)) \cap \left(\Downarrow_{\mathcal{V}\psi}^{\mathcal{V}(\varphi\wedge\psi)}\right)^{-1}(\mathscr{L}_V(\psi)) \\ \mathscr{L}_V(\exists X.\ \varphi) &\equiv \mathscr{L}_V(\varphi) \downarrow X \end{aligned}$$

Proof. The first equivalence is by definition of negation.

In the second equivalence we need $\Downarrow_{\mathcal{V}\varphi}^{\mathcal{V}\varphi\wedge\psi}$ to align $\mathcal{V}\varphi$ with $\mathcal{V}(\varphi\wedge\psi)$. The preimage of $\mathscr{L}_V(\varphi)$ allows free variables of ψ but not of φ to be assigned arbitrarily. The intersection of both preimages precisely implements conjunction.

For the last equivalence, we make a case distinction whether X is in $\mathcal{V}\varphi$. If X is not in $\mathcal{V}\varphi$ we have $\exists X.\varphi \equiv \mathscr{L}_V(\varphi) \equiv \mathscr{L}_V(\varphi)\downarrow X$.

Otherwise X is in $\mathcal{V}\varphi$ and let $\sigma \in \mathscr{L}_V(\exists X.\ \varphi)$. Then there is an M such that $\sigma_{X:=M} \vDash_0 \varphi$ (σ gets implicitly unfolded to an interpretation). Hence, $\sigma \otimes M \in \mathscr{L}_V(\varphi)$ and $\sigma \in \mathscr{L}_V(\varphi)\downarrow X$. For the other direction, let $\sigma \in \mathscr{L}_V(\varphi)\downarrow X$. Then there is an M such that $\sigma \otimes M \in \mathscr{L}_V(\varphi)$. Hence, $\sigma_{X:=M} \vDash_0 \varphi$, $\sigma \vDash_0 \exists X.\ \varphi$, and $\sigma \in \mathscr{L}_V(\exists X.\ \varphi)$. □

Lemma 8.6. *One can construct NFAs $\mathcal{A}_{X\lessdot Y}$ and $\mathcal{A}_{X\subseteq Y}$ such that*

$$\mathscr{L}_{\mathscr{A}}(\mathcal{A}_{X\lessdot Y}) \equiv \mathscr{L}_V(X \lessdot Y) \text{ and}$$
$$\mathscr{L}_{\mathscr{A}}(\mathcal{A}_{X\subseteq Y}) \equiv \mathscr{L}_V(X \subseteq Y).$$

Proof. Let $\Sigma := \mathbb{B}^{[X;Y]}$. Its easy to see that

$$\mathscr{L}_V(X \lessdot Y) \equiv \mathscr{A}(\Sigma^*\{a|a_X = \mathsf{true}\}\Sigma^*\{a|a_Y = \mathsf{true}\}\Sigma^\omega) \text{ and}$$
$$\mathscr{L}_V(X \subseteq Y) \equiv \mathscr{A}(\{a|a_X = \mathsf{true} \to a_Y = \mathsf{true}\}^\omega).$$

The languages Σ^*, $\{a|a_X = \mathsf{true}\}$, $\{a|a_Y = \mathsf{true}\}$, and $\{a|a_X = \mathsf{true} \to a_Y = \mathsf{true}\}$ are regular and we can construct NFAs accepting them. By Lemmas 5.5 and 5.7 and definition of $\mathscr{L}_{\mathscr{A}}(\mathcal{A})$ we can construct NFAs $\mathcal{A}_{X\lessdot Y}$ and $\mathcal{A}_{X\subseteq Y}$ precisely accepting the desired languages. □

Theorem 8.7. *For every formula φ one can construct an NFA $\mathcal{A}_\varphi$ with $\mathscr{L}_{\mathscr{A}}(\mathcal{A}_\varphi) \equiv \mathscr{L}_V(\varphi)$.*

Proof. We construct the NFA inductively on φ.

- For $X \lessdot Y$ and $X \subseteq Y$ we use the NFAs of Lemma 8.6.
- Given $\mathcal{A}_\varphi$ and $\mathcal{A}_\psi$, by Lemma 8.5

$$\mathcal{A}_{\varphi\wedge\psi} := \left(\Downarrow_{\mathcal{V}\varphi}^{\mathcal{V}\varphi\wedge\psi}\right)^{-1}(\mathcal{A}_\varphi) \cap \left(\Downarrow_{\mathcal{V}\psi}^{\mathcal{V}\varphi\wedge\psi}\right)^{-1}(\mathcal{A}_\psi)$$

  suffices.
- Given $\mathcal{A}_\varphi$, $\mathcal{A}_{\neg\varphi} := \mathcal{A}_\varphi^C$ suffices by Lemma 8.5.
- Given $\mathcal{A}_\varphi$, let $\mathcal{A}_{\exists X.\ \varphi} := \left(\Downarrow_{\mathcal{V}\varphi\setminus X}^{\mathcal{V}\varphi}\right)(\mathcal{A}_\varphi)$. This is correct by Lemmas 8.5 and 8.4 because $\mathscr{L}_V(\mathcal{A}_\varphi)$ is extensional.

Because $\mathscr{A}$ is admissible, we have correctness of all used closure operations on Büchi automata. □

Corollary 8.8. *Satisfiability of* MSO_0 *formulas is decidable.*

Proof. By Theorem 8.3 we can decide whether there is a satisfying vector interpretation. Because $\mathscr{A}$ is admissible, $\mathscr{L}_{\mathscr{A}}(\mathcal{A}_\varphi) \equiv \emptyset$ or $\exists \sigma \in \mathscr{L}_{\mathscr{A}}(\mathcal{A}_\varphi)$ is decidable. □

Corollary 8.9. *Satisfaction in* MSO_0 *is logically decidable.*

Proof. Let φ be a formula and J and interpretation. We need to show $J \vDash_0 \varphi \vee J \nvDash_0 \varphi$. This is equivalent to $\bigotimes_{\mathcal{V}\varphi} J \in \mathscr{L}_{\mathscr{A}}(\mathcal{A}_\varphi) \vee \bigotimes_{\mathcal{V}\varphi} J \notin \mathscr{L}_{\mathscr{A}}(\mathcal{A}_\varphi)$ by Theorem 8.3 and Theorem 8.7. Corollary 7.7 completes the proof. □

Note that the two prior theorems hold for all AS structures. We look at the special cases of the UP and the ω-structure.

Corollary 8.10. *In the UP-structure, satisfiability of* MSO_0 *formulas is decidable and satisfaction in* MSO_0 *is logically decidable.*

Corollary 8.11. *In the* ω*-structure, satisfiability of* MSO_0 *formulas is decidable and satisfaction in* MSO_0 *is logically decidable given* AR*.*

Lemma 8.12. *Given* AR*, an* MSO_0 *formula is satisfiable in the UP-structure if and only if it is satisfiable in the* ω*-structure.*

Proof. By Theorem 8.7 we need to prove that there is a $\sigma \in \mathscr{L}_B(\mathcal{A}_\varphi)$ if and only if there is a $xy^\omega \in \mathscr{L}_B(\mathcal{A}_\varphi)$, which follows from Corollary 4.22. □

Lemma 8.13. *Satisfaction in* MSO_0 *in the UP-structure is decidable.*

Proof. To decide $J \vDash_0 \varphi$ it suffices to decide $\bigotimes_{\mathcal{V}\varphi} J \in \mathscr{L}_{\mathsf{UP}}(\mathcal{A}_\varphi)$ by Theorem 8.7. Because $\bigotimes_{\mathcal{V}\varphi} J$ is guaranteed to be UP, the later is decidable by Lemma 5.8. □

8.3 MSO With First Order Variables

We enrich MSO_0 with first order variables and obtain MSO, to which one usually refers with S1S. MSO formulas are given by

$$\varphi, \psi ::= x < y \mid x \in X \mid X \subseteq Y \mid \neg\varphi \mid \varphi \wedge \psi \mid \exists x.\ \varphi \mid \exists X.\ \psi$$

The variables $x, y \in \mathbb{N}$ are first order variables ranging over numbers and $X, Y \in \mathbb{N}$ are second order variables ranging over sets of number. Sets are $\mathscr{A}$-sequences over $\mathbb{B}$ as before. A **first order interpretation** is a function from $\mathbb{N}$ to $\mathbb{N}$ and a **second order interpretation** is a function

from $\mathbb{N}$ to $\mathscr{A}(\mathbb{B})$. We use the letter I for first order and the letter J for second order interpretations. The satisfaction relation is defined as expected:

$$\begin{aligned} I, J \vDash x < y &:= Ix < Iy \\ I, J \vDash x \in X &:= (Ix) \in (JX) \\ I, J \vDash X \subseteq Y &:= \forall n.\ n \in (IX) \to n \in (JY) \\ I, J \vDash \neg\varphi &:= I, J \nvDash \varphi \\ I, J \vDash \varphi \wedge \psi &:= I, J \vDash \varphi \wedge I, J \vDash \psi \\ I, J \vDash \exists x.\ \varphi &:= \exists n.\ I_{x:=n}, J \vDash \varphi \\ I, J \vDash \exists X.\ \varphi &:= \exists M.\ I, J_{X:=M} \vDash \varphi \end{aligned}$$

For a reduction from MSO to MSO_0 we need to eliminate first order variables. First order variables will be represented by second order variables, for which we ensure that they are interpreted with singleton sets. We write $\{n\} := \mathsf{false}@^n\mathsf{true}$ for the singleton set containing n. Being a singleton can be expressed in MSO_0 as

$$\mathsf{sing}\ X := \neg X \lessdot X \wedge \exists Y.\ X \lessdot Y$$

where Y is different from X e.g. $Y := X + 1$.

Fact 8.14. *If $J \vDash_0 \mathsf{sing}\ X$, then one can construct an n such that $JX \equiv \{n\}$.*

Proof. By the right conjunct of $\mathsf{sing}\ X$ there is an $n \in JX$. Because $\in$ is decidable, one obtains an $n \in JX$ with constructive choice. By the left conjunct, n is the only number contained in JX. □

We will give four functions:

1. $\lfloor\cdot\rfloor$ translates an MSO formula into an MSO_0 formula.

2. $\lfloor\cdot,\cdot\rfloor$ translates a first order and a second order interpretation to an interpretation for MSO_0.

3. $\lceil\cdot\rceil_1$ translates an MSO_0 interpretation into a first order interpretation.

4. $\lceil\cdot\rceil_2$ translates an MSO_0 interpretation into a second order interpretation.

The functions should satisfy the follwing: if $I, J \vDash \varphi$ then $\lfloor I, J\rfloor \vDash_0 \lfloor\varphi\rfloor$ and if $J \vDash_0 \lfloor\varphi\rfloor$ then $\lceil J\rceil_1, \lceil J\rceil_2 \vDash \varphi$. Satisfaction and satisfiability of MSO can be translated to MSO_0 with these functions. We start with the translation functions for interpretations. The function $\lfloor\cdot,\cdot\rfloor$ merges two interpretations

into one. We use even variables in MSO_0 for first order variables in MSO and odd variables in MSO_0 for second order variables in MSO.

$$\begin{aligned} \lfloor x \rfloor &:= x + x \\ \lfloor X \rfloor &:= X + X + 1 \\ \lfloor I, J \rfloor X &:= \begin{cases} \{I(X\%2)\} & \text{if } X \text{ is even} \\ J(X\%2) & \text{if } X \text{ is odd} \end{cases} \end{aligned}$$

where $\%$ stands for down-rounding division on $\mathbb{N}$.

Fact 8.15. *$\lfloor I, J \rfloor \lfloor x \rfloor \equiv \{Ix\}$ and $\lfloor I, J \rfloor \lfloor X \rfloor \equiv JX$ for any first order variable x and second order variable X.*

When translating MSO formulas to MSO_0 formulas one needs to ensure that the variables used for free first order variables must be interpreted as singleton sets:

$$\begin{aligned} \lfloor x < y \rfloor &:= \lfloor x \rfloor \lessdot \lfloor y \rfloor \wedge \mathsf{sing}\ \lfloor x \rfloor \wedge \mathsf{sing}\ \lfloor y \rfloor \\ \lfloor x \in X \rfloor &:= \lfloor x \rfloor \subseteq \lfloor X \rfloor \wedge \mathsf{sing}\ \lfloor x \rfloor \\ \lfloor X \subseteq Y \rfloor &:= \lfloor X \rfloor \subseteq \lfloor Y \rfloor \\ \lfloor \neg \varphi \rfloor &:= \neg \lfloor \varphi \rfloor \\ \lfloor \varphi \wedge \psi \rfloor &:= \lfloor \varphi \rfloor \wedge \lfloor \psi \rfloor \\ \lfloor \exists x.\ \varphi \rfloor &:= \exists \lfloor x \rfloor.\ \mathsf{sing}\ \lfloor x \rfloor \wedge \lfloor \varphi \rfloor \\ \lfloor \exists X.\ \varphi \rfloor &:= \exists \lfloor X \rfloor.\ \lfloor \varphi \rfloor \end{aligned}$$

Lemma 8.16. *$I, J \vDash \varphi$ if and only if $\lfloor I, J \rfloor \vDash_0 \lfloor \varphi \rfloor$.*

Proof. By induction on φ using Fact 8.15, the fact that $\{n\} \lessdot \{m\}$ is equivalent to $n < m$, and $\{n\} \subseteq M$ is equivalent to $n \in M$. □

Fact 8.17. *If $J \vDash_0 \lfloor \varphi \rfloor$, then $J \vDash_0 \mathsf{sing}\ \lfloor x \rfloor$ for all free first order variables x in φ.*

Proof. By induction on φ. □

To give the function translating an MSO_0 interpretation to a first and a second order interpretation for MSO, fix a formula φ and assume that $J \vDash_0 \lfloor \varphi \rfloor$. The conversion to second order is easy. But only first order variables free in φ are guaranteed to be singleton sets.

$$\begin{aligned} \lceil J \rceil_1 x &:= \begin{cases} n & \text{if } x \text{ is free in } \varphi \text{ and } J\lfloor x \rfloor \equiv \{n\} \\ 0 & \text{otherwise} \end{cases} \\ \lceil J \rceil_2 X &:= J\lfloor X \rfloor \end{aligned}$$

Facts 8.14 and 8.17 justify the first case of $\lceil J \rceil_1$, because n can be constructed.

Lemma 8.18. *$J \vDash_0 \lfloor\varphi\rfloor$ if and only if $\lceil J\rceil_1, \lceil J\rceil_2 \vDash \varphi$.*

Proof. By induction on φ using Facts 8.14 and 8.17. □

Lemma 8.19. *Satisfaction in* MSO *is logically decidable.*

Proof. By Lemma 8.16 it suffices to show $\lfloor I, J\rfloor \vDash_0 \lfloor\varphi\rfloor \vee \lfloor I, J\rfloor \nvDash_0 \lfloor\varphi\rfloor$, which follows from Corollary 8.9. □

Lemma 8.20. *Satisfiability of* MSO *formulas is decidable.*

Proof. By Corollary 8.8 satisfiability of $\lfloor\varphi\rfloor$ is decidable. Assume there is a J satisfying $\lfloor\varphi\rfloor$. Then $\lceil J\rceil_1$ and $\lceil J\rceil_2$ satisfy φ by Lemma 8.18. Assume $\lfloor\varphi\rfloor$ is unsatisfiable. To show that φ is unsatisfiable, assume there are I and J satisfying φ. This contradicts that $\lfloor\varphi\rfloor$ is unsatisfiable using Lemma 8.16. □

The two prior theorems hold for all AS structures. Is easy to translate the relationship between the UP-structure and the ω-structure from MSO_0 to MSO. Then AR implies SL:

Definition 8.21 (SL)**.** *SL is the following proposition: Satisfaction in* MSO *in the ω-structure is logically decidable. Formally*

$$\mathsf{SL} := \forall\varphi IJ.\ I, J \vDash \varphi \vee I, J \nvDash \varphi,$$

where φ is an MSO *formula, I a first order interpretation, and J a second order interpretation of the ω-structure.*

Corollary 8.22. *AR implies SL.*

Proof. Follows by Lemma 8.19 because under AR the ω-structure is admissible. □

Corollary 8.23. *For the UP-structure, satisfaction in* MSO *is decidable. Given AR, an* MSO *formula is UP-satisfiable if and only if it is ω-satisfiable.*

Proof. By Lemmas 8.12 and 8.13 using the translation from MSO to MSO_0. □

Chapter 9

Necessity of Additive Ramsey

By now, we have shown that AR is equivalent to RF, that BC is equivalent to BU, that AR implies BC, and that AR implies SL. We want to show that AR (or equivalently RF) is necessary to show BC and SL. We are going to show that BC implies RF, that SL implies RF, and hence that AR, RF, BC, BU, and SL are equivalent. A collection of the formal definitions of the five propositions can be found in the appendix.

9.1 BC Implies RF

Let $(\Gamma, +)$ be a finite semigroup. We give a Büchi automaton accepting the ω-sequences admitting a Ramseyan factorization and prove that this Büchi automaton accepts all ω-sequences.

Lemma 9.1. *There is an NFA $\mathcal{A}_a$ accepting σ if and only if σ admits a Ramseyan factorization g and $\sum \sigma[g0..g1) = a$.*

Proof. We build an NFA $\mathcal{A}'_a$ which accepts the language $\{x | x \neq \varepsilon \wedge \sum x = a\}$. Let $\mathcal{A}'_a := (\Gamma + \{1,2\}, \{1\}, \{2\}, \rightarrow)$ where

$$1 \xrightarrow{b} d := b = d$$
$$c \xrightarrow{b} d := c + b = d$$
$$c \xrightarrow{b} 2 := c + b = a$$
$$1 \xrightarrow{b} 2 := b = a.$$

The NFA stores the current sum of the already processed part of the string in its state and is allowed to enter 2 if this sum is a. Choose $\mathcal{A}_a := \mathcal{A}_{\text{univ}} \cdot {\mathcal{A}'_a}^{\omega}$, where $\mathcal{A}_{\text{univ}}$ is an NFA accepting Γ^*.

Let $\sigma \in \mathscr{L}_B(\mathcal{A}_a)$. Then $\sigma \equiv x \cdot \Pi y_i$ and $\sum y_i = a$ for all i by construction of $\mathcal{A}_a$. Now the recursive function g with $g0 = |x|$ and $g(i+1) = gi + |y_i|$ is a Ramseyan factorization for σ because $\sum \sigma[gi..g(i+1)) = \sum y_i = a$.

Let σ admit a Ramseyan factorization g with $\sum \sigma[g0..g1) = a$. Then $\sigma \equiv \sigma[0..g0) \cdot \Pi\lambda i.\sigma[gi..g(i+1))$ and $\sum \sigma[gi..g(i+1)) = a$. Hence $\sigma \in \mathscr{L}_B(\mathcal{A}_a)$. □

Lemma 9.2. *There is an NFA $\mathcal{A}$ accepting σ if and only if σ admits a Ramseyan factorization.*

Proof. By Lemma 9.1 the NFA $\bigcup_{a\in\Gamma} \mathcal{A}_a$ suffices. □

Lemma 9.3. *BC implies RF.*

Proof. Let $\mathcal{A}$ be the NFA from Lemma 9.2. Using BC we have $\sigma \in \mathscr{L}_B(\mathcal{A})$ or $\sigma \notin \mathscr{L}_B(\mathcal{A})$. If $\sigma \in \mathscr{L}_B(\mathcal{A})$ then σ admits a Ramseyan factorization. If $\sigma \notin \mathscr{L}_B(\mathcal{A})$ then $\sigma \in \mathscr{L}_B(\overline{\mathcal{A}})$ by BC and there is an $xy^\omega \in \mathscr{L}_B(\overline{\mathcal{A}})$ by Corollary 4.22. But by Fact 3.2 xy^ω admits a Ramseyan factorization, hence $xy^\omega \in \mathscr{L}_B(\mathcal{A})$ contradicting $xy^\omega \in \mathscr{L}_B(\overline{\mathcal{A}})$. □

Thus AR, RF, BC, and BU are equivalent and AR implies SL. It remains to show that SL implies RF.

9.2 SL Implies RF

Let $(\Gamma, +)$ be a finite semigroup and σ be a fixed but arbitrary ω-sequence over Γ. Recall the proof that excluded middle implies RF in Section 3.2. It suffices to show $\mathsf{IPP} \vee \mathsf{DM}(\neg\mathsf{IPP})$ and $\mathsf{INF}_{\simeq} \vee \mathsf{DM}(\neg\mathsf{INF}_{\simeq})$ to show RF (recall the definitions of IPP and $\mathsf{INF}_{\simeq}$ in Definition 3.7). These two propositions can be proven using SL. Because we assume SL, we are allowed to make use of all logical connectives, have De Morgan laws for quantifiers in MSO formulas and hence can use all-quantifiers both for first and second order variables in MSO.

We want to encode the proposition $\sum \sigma[i..j) = c$ into an MSO satisfaction problem. It is important that the bounds of the sum are not fixed numbers, which would allow building a formula recursively, but should be MSO variables. To encode σ we use distinct variables X_a for each $a \in \Gamma$. The sequence σ is encoded into a second order interpretation J_σ assigning the set of all positions where a occurs in σ to X_a:

$$J_\sigma := \lambda X. \begin{cases} \lambda n.\text{if } \sigma n = a \text{ then } \mathsf{true} \text{ else } \mathsf{false} & \text{if } X = X_a \\ \emptyset & \text{otherwise} \end{cases}$$

We quantify over all intermediate sums $\sum \sigma[i..i+1)$ to $\sum \sigma[i..j)$ and ensure that they are correct. We use distinct variables Y_a for all $a \in \Gamma$ different from all X_a to encode the the intermediate sums.

We need formulas expressing $\leq$ and the successor relation:

$$x \leq y := \neg(y < x)$$
$$y \text{ succof } x := x < y \wedge \neg\exists z.\ x < z < y$$

We use MSO variables x for i and y for j and want to give a formula $\varphi^c_{x,y}$ such that $I, J_\sigma \vDash \varphi^c_{x,y}$ if and only if $\sum \sigma[Ix..Iy) = c$ for $Ix < Iy$. We split the definition into four parts: φ_{unique} states that a number cannot be in more than one Y_a, φ_{init} states that $\sum \sigma[i..i+1) = \sigma i$, φ_{step} that $\sum \sigma[i..k+1) = \sum \sigma[i..k) + \sigma k$ for $i < k < j$, and φ_{end} that $\sum \sigma[i..j) = c$. The formulas force the number k to be in Y_a if $\sum \sigma[i..k+1) = a$. Note that there is one intermediate sum less than colors to add and hence we need to use the predecessor of y in φ_{end}.

$$\varphi_{unique} := \forall z.\ \bigwedge_{a\in\Gamma} \left(z \in Y_a \rightarrow \bigwedge_{b\neq a} z \notin Y_b \right)$$

$$\varphi_{init} := \bigwedge_{a\in\Gamma} (x \in X_a \rightarrow x \in Y_a)$$

$$\varphi_{step} := \forall z.\ x \leq z < y \rightarrow \left(\bigwedge_{a\in\Gamma} z \in Y_a \rightarrow \left(\forall z'.\ z' \text{ succof } z \rightarrow \bigwedge_{b\in\Gamma} (z' \in X_b \rightarrow z' \in Y_{a+b}) \right) \right)$$

$$\varphi_{end} := \forall z.\ y \text{ succof } z \rightarrow z \in Y_c$$

$$\varphi^c_{x,y} := \exists\!\!\!\!\!\!\cdot \bigcup_{a\in\Gamma} Y_a.\ \varphi_{unique} \wedge \varphi_{init} \wedge \varphi_{step} \wedge \varphi_{end}$$

Lemma 9.4. *For any first order interpretation I with $Ix < Iy$, $I, J_\sigma \vDash \varphi^c_{x,y}$ if and only if $\sum \sigma[Ix..Iy) = c$.*

Proof. The formula φ_{unique} ensures that a number is in at most one of the Y_a. We prove that $n \in Y_a$ if and only if $\sum \sigma[Ix..n+1) = a$ for $Ix \leq n < Iy$ by induction.

If $Ix = n$, then φ_{init} and φ_{unique} ensure the claim. If $Ix < n+1 < Iy$, we need to show that $n+1 \in Y_a$ if and only if $\sum \sigma[Ix..n+2) = a$. By induction hypothesis $n \in Y_b$ if and only if $\sum \sigma[Ix..n+1) = b$. When choosing z as n in φ_{step}, φ_{step} and φ_{unique} require that $n+1 \in X_d$ if and only if $n+1 \in Y_{b+d}$. Because $a = \sum \sigma[Ix..n+2) = \sum \sigma[Ix..n+1) + \sigma(n+1) = b + d$ we have $n+1 \in Y_a$ if and only if $\sum \sigma[Ix..n+2) = a$.

Choosing n as $Iy-1$ yields that $Iy-1 \in Y_a$ if and only if $\sum \sigma[Ix..Iy) = a$. The formula φ_{end} ensures that $Iy - 1 \in Y_c$ and the lemma is proven. □

Using $\varphi^c_{x,y}$ we prove the propositions required to derive RF.

Lemma 9.5. *SL implies* $\mathsf{INF}_{\simeq} \vee \mathsf{DM}(\neg\mathsf{INF}_{\simeq})$.

Proof. By the definition of $\simeq$ (cf. Section 3.2) and Lemma 9.4, $\mathsf{INF}_{\simeq}$ is equivalent to the satisfaction problem

$$I, J_\sigma \vDash \exists x.\ \forall y.\ \exists z.\ y < z \wedge \exists z'.\ z < z' \wedge \bigvee_{a \in \Gamma} \varphi^a_{x,z'} \wedge \varphi^a_{z,z'}$$

for any I (because the formula has no free first order variables). By SL the formula is satisfied or not. If it is, then $\mathsf{INF}_{\simeq}$ holds. Otherwise $\mathsf{DM}(\neg\mathsf{INF}_{\simeq})$ holds by applying De Morgan laws to the formula (justified by SL). □

Lemma 9.6. *SL implies* $\mathsf{IPP} \vee \mathsf{DM}(\neg\mathsf{IPP})$.

Proof. The proof is analogous to Lemma 9.5. We encode IPP as an equivalent satisfaction problem:

$$I, J_\sigma \vDash \bigvee_{a \in \Gamma} \forall x.\ \exists y.\ x < y \wedge y \in X_a$$

for any I. By SL the formula is satisfied. If it is, IPP holds. Otherwise $\mathsf{DM}(\neg\mathsf{IPP})$ holds by applying De Morgan laws. □

Theorem 9.7. *SL implies RF.*

Proof. By Lemma 3.15 it suffices to show IPP and $\mathsf{INF}_{\simeq}$. By Lemmas 3.11 and 3.13, $\mathsf{IPP} \vee \mathsf{DM}(\neg\mathsf{IPP})$ and $\mathsf{INF}_{\simeq} \vee \mathsf{DM}(\neg\mathsf{INF}_{\simeq})$ suffice, which follow from Lemmas 9.5 and 9.6. □

Theorem 9.8. *AR, RF, BC, BU, and SL are equivalent.*

Proof. By Lemma 3.6 AR is equivalent to RF, by Theorem 7.18 BC is equivalent to BU, by Corollary 7.8 AR implies BC, by Corollary 8.22 AR implies SL, by Lemma 9.3 BC implies RF, and by Theorem 9.7 SL implies RF. Figure 9.1 illustrates the implications together with the proof ideas. □

Remark 9.9. *The implication from AR to SL could be split into an implication from AR to BC and from BC to SL. But then one has to modify Theorem 8.7 not to construct the NFA* $\mathcal{A}_\varphi$ *but only to show its existence. Hence, there must be another lemma constructing* $\mathcal{A}_\varphi$ *to show decidability of satisfiability.*

Corollary 9.10. *AR, RF, BC, BU, and SL are independent.*

Proof. RF is independent by Theorem 3.19 and thus are the others. □

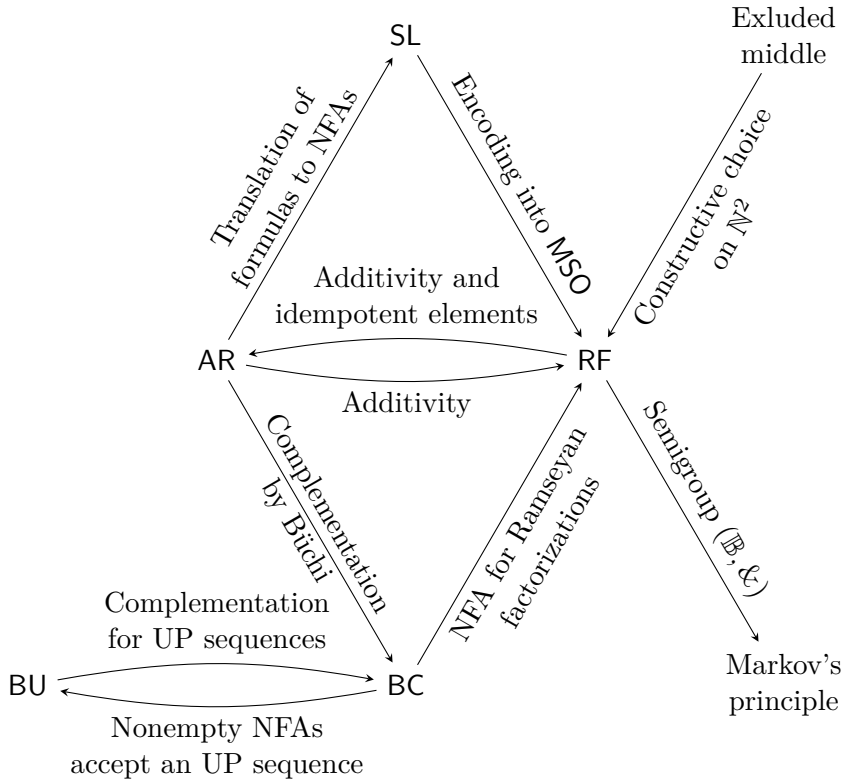


Figure 9.1: The implications between AR, RF, BC, BU, SL, excluded middle, and Markov's principle together with the major proof ideas.

Chapter 10

Related Work

We separate related work in two parts: Work related to our results and work related to the formalization of the results, where we make additional distinctions in the first group.

Complementation Complementation of Büchi automata was essential to translate formulas to Büchi automata. Out of the many different complementation approaches, we decided to follow the Ramsey-based complementation approach given by Büchi [3], Khoussainov and Nerode [10], Hofmann and Lange [9], and many others. All follow the same structure: they define the equivalence relation $\sim$, prove compatibility and totality, and show that the complement Büchi automaton exists. While Hofmann and Lange [9] build NFAs to show that the $\sim$ equivalence classes are regular and use Ramsey's Theorem to prove totality, Khoussainov and Nerode [10] use the Myhill-Nerode theorem for showing regularity of the $\sim$ equivalence classes and give a proof of totality without Ramsey's Theorem. We will come back to Khoussainov and Nerode's proof of totality later when discussing AR. It is not clear that other complementation constructions can be proven correct using AR only. For example, Safra [15] uses König's Lemma for his translation from (nondeterministic) Büchi automata to (deterministic) Rabin automata, which can be complemented easily.

UP Sequences Büchi automata accept exactly the ω-regular languages. ω-regular languages are usually defined using ω-regular expressions. As we have not formalized ω-regular expressions, we did not use this term in our work. Using (Additive) Ramsey to show that the vector languages of S1S formulas are ω-regular is uncritical classically because there Ramsey's Theorem holds. Hence, it is clear that is suffices to look for UP sequences when deciding satisfiability.

Calbrix et al. [5] show that an ω-regular language is fully determined by the UP sequences it contains. Additionally, Calbrix et al. [6] give a decision

procedure for satisfiability of S1S based on UP sequences and NFAs on strings. An UP sequence xy^ω can be encoded as the string $x\$y$ where \$ is a fresh symbol. Formulas are translated to NFAs accepting these strings. This translation differs from ours because we use Büchi acceptance even for UP sequences. We conjecture that the construction by Calbrix et al. can be verified constructively for UP sequences. To show that two ω-regular languages containing the same UP sequences are equivalent (BU), Calbrix et al. use complementation. We showed that this is constructively equivalent to BC and thus using NFAs on strings for UP sequences cannot give a stronger result than decidability of S1S in the UP-structure.

Bresolin et al. [2] study Büchi automata that only accept UP sequences. Their approach is different from ours. We only care about UP sequence when studying Büchi automata in the UP-structure and are not interested in the behavior of a Büchi automaton on other sequences. Bresolin et al. restrict the underlying NFAs such that they can only accept UP sequences but no others. These restricted NFAs are not closed under complementation because the complement of a language only containing UP sequences contains all sequences which are not UP.

Additive Ramsey In the paper on decidability of S1S, Büchi [3] uses the unrestricted version of Ramsey's Theorem. The restriction to additive colorings seems necessary when Büchi [4] generalizes from $\mathbb{N}$ to ω_1, even Büchi did not use the term "additive". He already uses the equivalence relation $\simeq$ we used to derive AR from excluded middle. The name "additive" and a clear encapsulation of Additive Ramsey's Theorem is given by Shelah [17], where Shelah generalizes to ω_2 without using automata. He also uses the $\simeq$ relation.

Ramseyan factorizations occur for example in Perrin and Pin [14] as in Blumensath [1]. Blumensath uses Ramseyan factorizations to translate S1S formulas to finite Wilke algebras. To our knowledge the equivalence of RF and AR was not acknowledged by now, even if they are closely related. Perrin and Pin as Blumensath show RF using the unrestricted version of Ramsey's Theorem or prove it in the same way the unrestricted Ramsey's Theorem is proven. Classically, the proof of Ramsey's Theorem is very elegant, but from a constructive point of view unnecessarily strong. The proof of AR by Shelah and the one of totality by Khoussaniov and Nerode are comparable, because both are using the $\simeq$ equivalence. The later proof is more detailed and derives totality directly, where the extension to AR is easy. We started with this proof for showing that SL implies RF and ended up with a simplified version, which has parallels to a proof by Kołodziejczyk et al. [11] (see next paragraph for details).

Equivalence of AR, BC, and SL Kołodziejczyk et al. [11] show a reminiscent result to ours in the logic RCA_0. They show that in RCA_0 complementation of Büchi automata, decidability of S1S[1], AR, and induction on Σ^0_2 formulas are equivalent. The logic RCA_0 is very different from constructive type theory and plays an important role in reverse mathematics. RCA_0 is a classical logic with the law of excluded middle but with a limited strength of induction. RCA_0 admits only Σ^0_1 induction, which restricts the induction hypothesis to be of the form $\exists x_1 \ldots \exists x_n.\ \varphi$, where φ contains only bounded quantifiers ($\exists x < m.\ \psi$ or $\forall x < m.\ \psi$). The Σ^0_2 induction principle allows induction hypotheses of the form $\exists x_1 \ldots \exists x_n.\ \forall y_1 \ldots \forall y_m.\ \varphi$, where φ contains only bounded quantifiers again. Constructive type theory allows induction hypotheses with arbitrarily many quantifier alternations but has no excluded middle. This makes Σ^0_2 induction trivial in type theory. On the other hand, we show equivalence of AR to complementation and logically decidability of the word problem for Büchi automata (so to BC). In RCA_0, logical decidability follows directly from excluded middle. Another remarkable point is that Kołodziejczyk et al. show equivalence to decidability of S1S and not to logical decidability of satisfaction in S1S (SL), which in RCA_0 is again trivial by excluded middle.

Although our proof that SL implies RF originated from Khoussainov and Nerode [10], we end up with a proof with parallels to the proof that Σ^0_2 induction implies AR. Kołodziejczyk et al. first prove Ordered Ramsey's Theorem (without an $\simeq$ equivalence). Ordered Ramsey's Theorem does not require a finite semigroup of colors, but a partial ordering on the colors and that the coloring is compatible with the order. Then they derive AR using semigroup theory from Ordered Ramsey's Theorem. To prove Ordered Ramsey's Theorem, Kołodziejczyk et al. build a similar sequence of pairs (n_i, k_i) as we do in Lemma 3.12.

Formalization Dozkal and Smolka [8] give a formalization of regular language representations in Coq including NFAs and WS1S. WS1S is the weak monadic second order logic, which restricts sets to be finite and hence they can are representanble by strings. Similar to us, they use dependently typed structures to represent NFAs. The translation from WS1S to NFAs corresponds to the translation from S1S to Büchi automata. The translation to Büchi automata is easier, because one has not to deal with strings of different length. Dozkal and Smolka use a less than relation $N \lessdot M$ on sets, too. Where we require an $n \in N$ and $m \in M$ with $n < m$ to exist, they require all $n \in N$ to be less than all $m \in M$. The existential quantifier is critical for singleton sets on sequences because only then one can obtain the number

[1] They show equivalence to the depth-n fragment of S1S for any $n \geq 5$, because satisfaction cannot be expressed in RCA_0 for an arbitrary formula without known quantifier rank.

in the singleton using constructive choice. The formalization of NFAs on strings is more compact than on sequences. Because strings are lists defined inductively, acceptance can be defined inductively as well, which results in an elegant Coq formalization. Because we represent sequences by functions, our formalization lacks this elegance. The formalization by Dozkal and Smolka uses Ssreflect and hence another implementation of e.g. finite types, which makes it difficult for us to use their results for regular languages in our formalization.

ω-languages do not to play an important role in formalization in proof assistants. Schimpf et al. [16] give a verified decision procedure for LTL model checking in Isabelle/HOL. LTL formulas are translated to generalized Büchi automata. This translation is not structurally recursive on the LTL formulas and no closure properties of (generalized) Büchi automata are required. Merz [13] gives a formalization of weak alternating automata. The formalization is provided in Isabelle/HOL. He proves closure under complement of weak alternating automata following Thomas [21] using infinite games. Sequences are formalized as function as in our development. The formalization contains no proof that Büchi and weak alternating automata accept the same class of languages.

Chapter 11

Remarks on Coq Development

All results presented here are fully formalized following the organization of this thesis. We discuss some problem in formalizing this work and argue why some aspects are formalized differently as presented here.

The formalization depends heavily on the use of finite types, since they occur almost everywhere. We use the implementation of finite types by Menz [12], which was usable nicely in general but which has some technical extensions needed for vector types. In Chapter 8 we identified vector types $\mathbb{B}^{l_1}$ and $\mathbb{B}^{l_2}$ for lists l_1 and l_2 to obtain compact definitions, e.g. for the outprojection of a variable in Section 8.2. This type identification is not possible in Coq. Hence, we need to use converter functions and definitions as proofs get more technical.

Defining sequences as functions $\mathbb{N} \to \Sigma$ involves a lot of arithmetic on natural numbers as the definition of infinitely often as $\forall n.\ \exists m \leq n.\ pm$ does. Arithmetic is not elegant in Coq and hence some proofs get much longer than one would hope for. The tactic `omega` for solving arithmetical goals is essential for doing these proofs. Still, one has to solve several equations manually. Especially when proving correctness of closure operations of Büchi automata, showing that a given Büchi automaton accepts a given language can be tedious and far away from intuition. `Setoid` rewriting with sequence and language equivalence is crucial to be able to work with sequences in Coq.

Alternatively, sequences could be defined as streams. Inductive definitions of lists and acceptance yield more elegant proofs regarding regular languages than for Büchi automata in our development. We are not sure whether streams would make things more elegant and representing decidable sets by streams over $\mathbb{B}$ could be more complicated.

Formalizing S1S and the translation to Büchi automata is easier than formalizing closure operations of Büchi automata because formulas are de-

	Specification	Proof
Preliminaries	522	1156
Ramseyan Properties	147	432
NFAs	238	491
Basic Operation on Büchi Automata	225	459
Büchi Complementation	182	551
AS Structures	156	453
S1S	487	934
Necessity of AR	167	473
Total	2123	4949

Figure 11.1: Size (lines of code) of our Coq Development divided according to the Chapters of this work.

fined inductively. Using the closure operations is simple. This hides the complexity of dealing with sequences in the closure operations of Büchi automata. We used named variables for our definition of S1S (where Dozkal and Smolka [8] used De Bruijn indices). Using names was not a problem in the formalization.

Our formal development differs from the paper version in some points. Strings are not represented by lists but by nonempty prefixes of sequences. A string is defined as a pair of a sequence and the last index of the sequence still belonging to the string. Hence, we cannot represent the empty string and do not have a usable equality on strings. On the other hand, index based access of symbols is simplified. Because strings actually consist of sequences, strings and sequences behave uniformly and proofs get simpler. Instead of equality on strings we use an equivalence as we already do for sequences. In the definition of the ω-concatenation and ω-iteration of languages it was crucial to use nonempty strings. Because we cannot represent the empty string, this becomes trivial and made definitions of these functions much easier. We do not have to give proofs to the functions, that strings are actually nonempty. On the other hand, the lack of the empty string requires more case distinctions when the empty string would be usable.

AS structures are represented as dependently typed structures similar to our definition in Chapter 7. For UP sequences, it is important that the representation as pairs of strings can be chosen. Then these strings become available for computation to decide whether an UP sequence is contained in the language of a Büchi automaton (Lemma 5.8) or to decide satisfaction in the UP-structure (Lemma 8.13). The necessity to define operations C, $\circ$, $\otimes$, and @ for each AS structure may seem unnecessarily complicated. It encapsulates that the ω-sequences, that can be represented by an AS structure, are closed under these operations when one sees an AS structure as a subset of Σ^ω.

The technicality of proofs regarding sequences becomes apparent in the comparison of lines of code for specification and proofs. Our development consists of about 7100 lines of code, where only 2100 are specification and the remaining 5000 are proofs. Note that Doczkal and Smolka [8] have a proportion of 1300 to 1700 lines of code. For more details see Figure 11.1. Apart from the library for finite types, we make use of Smolka [18] for an implementation of lists and basic decidability facts. The library for finite types by Menz [12] contains some code of Smolka and Stark [19] for turning decidable predicates into proof irrelevant predicates. The development can be found at `http://www.ps.uni-saarland.de/~lichter/master/`.

Chapter 12

Conclusion and Future Work

We have seen that the methods and constructions for proving decidability of S1S can be transferred from from classical logic to constructive type theory. We need to assume AR to show correctness, but not excluded middle. AR suffices to show decidability of MSO formulas and SL. As ω-sequences are infinite, we cannot reason completely constructively about sequences and need to use AR. But for the restricted UP-structure we prove the same constructions to be correct without assumptions.

Totality is the crucial property for complementing Büchi automata. Defining the complementation operation on Büchi automata is not a problem, but proving its correctness is not possible for ω-sequences constructively, because totality cannot be proven. Totality as in Lemma 6.14 cannot be proven constructively because for an arbitrary sequence σ we need to show that a language VW^ω containing σ exists. But this VW^ω depends on the behavior of σ towards infinity. With AS structures we separate totality from the rest of the proofs. This makes it easy to instantiate the results for ω-sequences under AR and for UP sequences constructively.

The remaining closure operations on Büchi automata can be proven correct following the classical proofs. Constructively, there are some critical points, for instance the Büchi acceptance condition. It is not equivalent constructively, whether there is a single final state occurring infinitely often in a sequence, or whether some final states occur infinitely often in the sequence. Classically, both conditions are equivalent and appear for instance in Thomas [21] and Vardi [24].

To show decidability of satisfiability and logical decidability of satisfaction, S1S formulas are translated to Büchi automata using closure operations. To obtain an easier translation, we first translate MSO_0 formulas without first order variables to Büchi automata and then MSO formulas with first order variables to MSO_0 formulas. This separation makes formal proofs a lot simpler. The AS structures transfer to the logic and we can have a constructive view on S1S using UP sequences or a more classical one

with ω-sequences under AR. Because AS structures are only needed for correctness proofs and not for constructions, showing equisatisfiability of both structures under AR is easy.

We restrict S1S to use decidable sets represented by sequences over $\mathbb{B}$. Allowing sets to be undecidable by using sequences over $\mathbb{P}$ would make it impossible to run Büchi automata on these sequences. The constructions for Büchi automata rely e.g. on the decidability of the transition relation. But this requires to know which symbol at some positions of a sequence occurs. Additionally the translation form MSO to MSO_0 would not be possible because we need constructive choice to obtain the member in a singleton set and constructive choice requires sets to be decidable. Using sequences over $\mathbb{B}$ for sets in S1S is justified becuase the same translation to Büchi automata as for S1S in classical logic with sets can be used and we are able to show similar results.

In retrospect, it becomes understandable that S1S with ω-sequences cannot be treated constructively but that we need something, in our case AR, which reduces the infinity of ω-sequences to something finite. It is not surprising that the restriction to UP sequences, which still have an infinite meaning but are represented finitely, allows constructive correctness proofs.

Last, AR is necessary for both, BC and SL. AR, RF, BC, BU and SL are equivalent. To show that SL implies RF, SL is strong enough to reuse the proof that excluded middle implies RF because the usages of excluded middle can be replaced by encodings into S1S and SL. The proof that BC implies RF is much more elegant and reveals benefits of nondeterminism. Showing the equivalence between BC and BU requires complementation for the UP-structure and gives an additional justification for using AS structures, even if one is only interested in results for the ω-structure. As RF is independent, all of the other propositions are independent as well.

When formalizing these results, we recognized that using ω-sequences as functions is much less pleasant in Coq than using inductive definitions for strings, but still this should not be a problem for experienced Coq users. In most cases one can follow the classical proofs. The increased effort may be grounded in sequences being not defined inductively, that number arithmetic is elaborate in Coq, that some problems on sequences get harder than on strings, and our design decisions.

Future Work

We gave two AS structures, the UP-structure and the ω-structure. Are there other AS structures? Are there fully constructive AS structures more expressive than UP sequences? By now, we were not able to find one. Is there an AS structure between the UP-structure and the ω-structure in the sense that is uses strictly weaker assumptions than AR, e.g. only the infinite

pigeonhole principle (see Definition 3.7)?

McNaugthon's theorem states that for every (nondeterministic) Büchi automaton there is a (deterministic) Muller automaton accepting the same language. It can be used to show that S1S is as expressive as weak S1S where quantification is only allowed over finite sets, but free set variables may still be infinite[1]. We do not expect that McNaugthon's theorem can be shown constructively, but it would be pleasant if one can show it using AR or even more, that it equivalent to it. This relates to the question whether different acceptance conditions for sequences, e.g. Büchi, Muller, Rabin, or Street acceptance, which are equivalent classically, are equivalent constructively as well.

Complementation is essential for the translation from formulas to Büchi automata and for showing SL. We used the Ramsey-based approach only depending on AR. Because AR and BC are equivalent, Ramsey-based complementation requires minimal assumptions. But there are different complementation constructions and which of them can be proven only using AR and which require more?

In this work we covered two representations of ω-regular languages, Büchi automata and S1S. Beside other acceptance conditions for NFAs, one could take a look at ω-semigroups and infinite games.

To show that S1S is decidable we needed AR but not full excluded middle. What happens when moving from S1S to S2S (monadic second order logic with two successors) or on the automata view from automata on sequences to automata on infinite trees? S2S was proven to be decidable by Rabin. Formulas can be translated to automata on infinite trees, for which emptiness is decidable. Under which assumptions can this be proven constructively?

[1]The usage of "weak" or WS1S sometimes refers to the monadic second order logic where all sets variables are finite or the monadic second order logic where only quantified variables needs to be finite. The first one is as expressive as regular languages, the second one, at least classically, as expressive as ω-regular languages.

Bibliography

[1] Achim Blumensath. *Monadic Second-Order Logic*. Tech. rep. Lecture Notes. Jan. 2015.

[2] Davide Bresolin, Angelo Montanari, and Gabriele Puppis. "A theory of ultimately periodic languages and automata with an application to time granularity". In: *Acta Inf.* 46.5 (2009), pp. 331–360.

[3] Julius Richard Büchi. "On a Decision Method in Restricted Second-Order Arithmetic". In: *International Congress on Logic, Methodology, and Philosophy of Science*. Stanford University Press, 1962, pp. 1–11.

[4] Julius Richard Büchi and Dirk Siefkes. *Decidable Theories: Vol. 2: The Monadic Second Order Theory of All Countable Ordinals*. Ed. by G.H. Müller and Dirk Siefkes. Lecture Notes in Mathematics. Springer Berlin Heidelberg, 1973.

[5] Hugues Calbrix, Maurice Nivat, and Andreas Podelski. "Ultimately Periodic Words of Rational w-Languages". In: *Mathematical Foundations of Programming Semantics, 9th International Conference, New Orleans, LA, USA, April 7-10, 1993, Proceedings*. 1993, pp. 554–566.

[6] Hugues Calbrix, Maurice Nivat, and Andreas Podelski. "Une méthode de décision de la logique mandique du second ordre d'une fonction successor". In: *Comptes rendus de l'académie des sciences, Serié I.* 318 (1994), pp. 847–850.

[7] Thierry Coquand and Bassel Mannaa. "The Independence of Markov's Principle in Type Theory". In: *1st International Conference on Formal Structures for Computation and Deduction (FSCD 2016)*. Ed. by Delia Kesner and Brigitte Pientka. Vol. 52. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, 17:1–17:18.

[8] Christian Doczkal and Gert Smolka. *Regular Language Representations in the Constructive Type Theory of Coq*. Tech. rep. submitted for review. Saarland University, Apr. 2017.

[9] Martin Hofmann and Martin Lange. *Automatentheorie und Logik*. Springer-Verlag Berlin Heidelberg, 2011.

[10] Bakhadyr Khoussainov and Anil Nerode. *Automata Theory and its Applications*. Birkhäuser, Boston, 2011.

[11] Leszek Aleksander Kolodziejczyk et al. "The Logical Strength of Büchi's Decidability Theorem". In: *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*. 2016, 36:1–36:16.

[12] Jan Christian Menz. *A Coq Library for Finite Types*. Bachelor Thesis, Saarland University. July 2016.

[13] Stephan Merz. "Weak Alternating Automata in Isabelle/HOL". In: *Theorem Proving in Higher Order Logics, 13th International Conference, TPHOLs 2000, Portland, Oregon, USA, August 14-18, 2000, Proceedings*. 2000, pp. 424–441.

[14] Dominique Perrin and Jean-Éric Pin. *Infinite Words - Automata, Semigroups, Logic and Games*. Elsevier, 2004.

[15] Shmuel Safra. "On the Complexity of omega-Automata". In: *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*. 1988, pp. 319–327.

[16] Alexander Schimpf, Stephan Merz, and Jan-Georg Smaus. "Construction of Büchi Automata for LTL Model Checking Verified in Isabelle/HOL". In: *Theorem Proving in Higher Order Logics, 22nd International Conference, TPHOLs 2009, Munich, Germany, August 17-20, 2009. Proceedings*. 2009, pp. 424–439.

[17] Saharon Shelah. "The Monadic Theory of Order". In: *Annals of Mathematics* 102.3 (1975), pp. 379–419.

[18] Gert Smolka. *Base Library for ICL*. Saarland University. 2016.

[19] Gert Smolka and Kathrin Stark. "Hereditarily Finite Sets in Constructive Type Theory". In: *Interactive Theorem Proving - 7th International Conference, ITP 2016, Nancy, France, August 22-27, 2016*. Ed. by Jasmin Christian Blanchette and Stephan Merz. Vol. 9807. LNCS. Springer, 2016, pp. 374–390.

[20] *The Coq Proof Assistant*. URL: `https://coq.inria.fr/`.

[21] Wolfgang Thomas. "Complementation of Büchi Automata Revised". In: *Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa*. 1999, pp. 109–120.

[22] Wolfgang Thomas. "Languages, Automata, and Logic". In: *Handbook of Formal Languages, Vol. 3*. Ed. by Grzegorz Rozenberg and Arto Salomaa. Springer-Verlag Berlin Heidelberg, 1997, pp. 389–455.

[23] Ming-Hsien Tsai et al. "State of Büchi Complementation". In: *Logical Methods in Computer Science* 10.4 (2014).

[24] Moshe Y. Vardi. "The Büchi Complementation Saga". In: *STACS 2007, 24th Annual Symposium on Theoretical Aspects of Computer Science, Aachen, Germany, February 22-24, 2007, Proceedings*. 2007, pp. 12–22.

Appendix

Important Propositions

The following propositions are equivalent:

Every sequence admits a Ramseyan factorization: (p. 23)

$\mathsf{RF} := \forall \sigma.\ \exists \text{ strictly montone } g.\ \forall i.\ \sum \sigma[g0..g1) = \sum \sigma[gi..g(i+1))$

Additive Ramsey: (p. 24)

$\mathsf{AR} := \forall \text{ additive } f.\ \exists \text{ strictly monotone } g.\ \forall i < j.\ f(g0)(g1) = f(gi)(gj)$

Büchi automata are closed under complement and the word problem is logically decidable: (p. 45)

$\mathsf{BC} := \forall \mathcal{A}.\ (\exists \overline{\mathcal{A}}.\ \mathscr{L}_B(\overline{\mathcal{A}}) \equiv \overline{\mathscr{L}_B(\mathcal{A})}) \wedge (\forall \sigma.\ \sigma \in \mathscr{L}_B(\mathcal{A}) \vee \sigma \notin \mathscr{L}_B(\mathcal{A}))$

The languages of Büchi automata accepting the same UP sequences are equivalent: (p. 56)

$\mathsf{BU} := \forall \mathcal{A}_1 \mathcal{A}_2.\ \mathscr{L}_{\mathsf{UP}}(\mathcal{A}_1) \equiv \mathscr{L}_{\mathsf{UP}}(\mathcal{A}_2) \to \mathscr{L}_B(\mathcal{A}_1) \equiv \mathscr{L}_B(\mathcal{A}_2)$

Satisfaction in S1S is logically decidable: (p. 66)

$\mathsf{SL} := \forall \varphi I J.\ I, J \vDash \varphi \vee I, J \nvDash \varphi$

Where σ is a sequence over Γ, f a function of type $\mathbb{N} \to \mathbb{N} \to \Gamma$, $(\Gamma, +)$ an arbitrary finite semigroup, $\mathcal{A}$, $\mathcal{A}_1$, and $\mathcal{A}_2$ are NFAs over some finite alphabet (without a semigroup operation), φ an MSO formula, I a first order interpretation, and J a second order interpretation of the ω-structure. Figure 9.1 on page 71 shows the implications between the prior propositions.

To show that the prior propositions are independent, we used the following propositions:

Excluded middle $:= \forall p.\ p \vee \neg p$

Markov's principle $:= \forall \sigma \in \mathbb{B}^\omega . \neg(\forall n.\ \sigma n = \mathsf{true}) \to \exists n.\ \sigma n = \mathsf{false}$ (p. 30)

Variables

We use the following letters for the following objects:

i, j, k, m, n	$\mathbb{N}$
Σ, Γ	alphabets
a, b, c, d	symbols
x, y, u, v	strings
σ, τ	sequences
$\mathcal{A}$	NFAs
q, p	states
r, s	finite runs
ϱ, ξ	infinite runs
x, y	first order variables
X, Y	second order variables
M, N	$\mathbb{N}$-sets
φ, ψ	formulas
I	first order interpretations
J	second order interpretations

Note that letters for strings and first order variables overlap, which is common.

List of Notations

Sequences

$x \cdot \sigma$ or $x\sigma$	prepending x to σ
$\sigma[n..]$	dropping the first n symbols of σ
$\sigma[n..m)$	substring of σ from n (inclusively) to m (exclusively)
$\sigma \equiv \tau$	sequence equivalence
$\sigma \circ f$	functional composition
$\sigma \otimes \tau$	product operation on sequences
$a@^n b$	sequence with b at position n and otherwise a
Πf or Πx_i	ω-concatenation of f or x_i
x^ω	ω-iteration of x
$\sigma \# p$	strictly monotone sequence of all positions of σ satisfying p
$\sigma \vert p$	ω-filter of σ with p

Languages

$L_1 \equiv L_2$	language equivalence
$f(L)$	image of L under f
$f^{-1}(L)$	preimage of L under f
$L \cdot L_\omega$ or LL_ω	L prepended to L_ω
L^ω	ω-iteration of L

Finite Semigroups

$(\Gamma, +)$	finite semigroup over type Γ with operation $+$
$\sum x$	sum of all elements in x

NFAs and Büchi Automata

$\mathscr{L}_R(\mathcal{A})$	regular language accepted by $\mathcal{A}$
$\mathscr{L}_R^+(\mathcal{A})$	language of nonempty strings accepted by $\mathcal{A}$
$\mathscr{L}_B(\mathcal{A})$	(Büchi) language accepted by an $\mathcal{A}$
$f^{-1}(\mathcal{A})$	Büchi automaton accepting the preimage
$f(\mathcal{A})$	Büchi automaton accepting the image
$\mathcal{A}_1 \cup \mathcal{A}_2$	Büchi automaton accepting the union
$\mathcal{A}_1 \cap \mathcal{A}_2$	Büchi automaton accepting the intersection
$\mathcal{A}_1 \cdot \mathcal{A}_2$	Büchi automaton obtained by prepending the NFA $\mathcal{A}_1$ to the Büchi automaton $\mathcal{A}_2$
$\mathcal{A}^\omega$	ω-iteration of an NFA
$\mathcal{A}^C$	complement Büchi automaton

Admissible Sequence Structures

$\mathscr{A}$	AS structure
$\mathscr{L}_{\mathscr{A}}(\mathcal{A})$	language of accepted $\mathscr{A}$-sequences by a Büchi automaton
$\mathscr{L}_{\mathsf{UP}}(\mathcal{A})$	language of accepted UP sequences by a Büchi automaton

S1S

$J \vDash_0 \varphi$	satisfaction in MSO_0
$\mathcal{V}\varphi$	free variables of an MSO_0 formula
$J_{X:=M}$	interpretation assigning X to M and all other variables according to J
$\mathscr{L}_V(\varphi)$	vector language of an MSO_0 formula
$\Downarrow_{l_2}^{l_1}$	projection of vectors of type $\mathbb{B}^{l_1}$ to vectors of type $\mathbb{B}^{l_2}$
$L \downarrow X$	outprojection of X from L
$I, J \vDash \varphi$	satisfaction in MSO