Diplomarbeit

Fragments of Temporal Logic and Formal Languages

Peter Lohmann

October 10, 2008

Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Theoretische Informatik

Abstract. We will investigate the relationships between classes of formal languages defined by various means. First we will show the equivalence of first order definable languages, where the defining formula may use the binary relations $<$ and successor and may only use two different variable names, to the languages definable by temporal logic formulas, with **X**, **Y**, **F** and **O** as the only temporal operators, and to the languages whose syntactic semigroup belongs to $\mathbf{DA} * \mathbf{D}$. We will show that the equivalence still holds if the successor relation is no longer used, only **F** and **O** are used and $\mathbf{DA} * \mathbf{D}$ is replaced with **DA**. All of this was already discussed in detail by Etessami, Vardi, Wilke [EVW97] and Thérien [TW98]. We will then show the equivalence of first order definability, with only two different variables and without the $<$ relation, to temporal logic definability, with only **X**, **Y** and the newly introduced **D**, and to local threshold testability with a threshold of 2.

Acknowledgments

I hereby would like to thank Heribert Vollmer for the supervision of this thesis and for his advice on topics suitable for research and worthwhile sources. Furthermore I would like to thank Thomas Zeume for proof-reading an earlier draft and pointing out possible improvements, especially in the formulation of some of the proofs. I thank Michael Thomas and Arne Meier for their help, I especially thank Arne for a fruitful discussion about the two different **D** operators. I would also like to thank my wife Anna for correcting linguistical mistakes and for her support. Most of all I want to thank God for his endless love and for giving me hope when the work on this thesis did not go forward as I had expected.

Contents

1 Introduction

Logics are a major field of theoretical computer science. One of those is (Linear) Temporal Logic which is similar to modal logic and was introduced by Pnueli [Pnu77] as a way to express properties of computing systems. Temporal Logic is an extension of propositional logic with several temporal operators which are used to describe the change of system state over time. Though not as expressive as general first order logic, temporal logic can be used to describe many typical requirements of computing systems. And, unlike in the case of general first order logic, checking whether a given system fulfills a given temporal logic formula is still computable – even in PSPACE.

Another major field of research in theoretical computer science is formal language theory. Within formal language theory the theory of regular languages – which form the lowest level of the Chomsky hierarchy of formal languages – forms a wide field of study in its own. The field is especially wide in the number of techniques commonly used to approach it, e.g. automata theory, combinatorics, logics or semigroup theory. In this thesis we will combine several of these approaches by proving some connections between them. We will characterize subclasses of regular languages by means of temporal logic, first order logic, combinatorics and semigroup theory and then show equalities between the hereby defined classes of languages.

On a larger scale the class of languages, which have a finite aperiodic syntactic semigroup, equals the class of languages, which can be defined by a first order formula over the vocabulary $\{<, \mathrm{succ}, P_0, P_1, P_2, \dots\}$, where $<$ is the usual order relation on the natural numbers, succ is the binary successor relation and the P_i are unary relations. The proof of this equality is due to Schützenberger [Sch65], McNaughton and Papert [MP71]. Kamp showed in [Kam68] that this class can also be characterized as the class of all languages which can be defined by a temporal logic formula.

On a smaller scale (which we will focus on) the class of languages, which can be defined by a temporal logic formula that does not use either of the two binary temporal operators **XU** and **YS**, was shown by Etessami, Vardi and Wilke [EVW97] to be equal to the class of languages, which can be defined by a first order formula which uses at most two different names for variables. Thérien and Wilke [TW98] showed that this class is equal to the class of languages, whose syntactic semigroup is in the class $\mathbf{DA} * \mathbf{D}$ which denotes the semidirect product of the semigroup pseudovarieties **DA** and **D**.

There are several natural subclasses of the latter class: One can restrict the vocabulary in the first order characterization by only allowing one out of $<$ or succ. Or one can forbid some more temporal operators in the temporal logic characterization, e.g. only use **X** and **Y** or only use **F** and **O**.

In [EVW97] it was also shown that the class, obtained by only allowing $<$ and not using succ in the first order logic characterization, equals the class, obtained by using **F**

and **O** as the only temporal operators in the temporal logic characterization. In [TW98] it was shown that this class equals the class of languages whose syntactic semigroup is in **DA**.

So the question remains whether the class obtained by only using succ and the class obtained by only using **X** and **Y** are equal. However, this can be very easily falsified as only using **X** and **Y** leads to a class which is even too small to define all languages definable by first order formulas with neither $<$ nor succ but only the unary relations.

But we will nonetheless show two other characterizations of the class, obtained by only allowing succ, not using $<$ and not using more than two different variables. The first one is a temporal logic characterization which follows almost immediately from the work in [EVW97]: The latter class is equal to the class, obtained when using **X**, **Y** and **D** as the only temporal operators, hereby **D** is not one of the standard temporal operators but newly introduced in this thesis. We will then look into a combinatorial characterization called local threshold testability. The class of all locally threshold testable languages is long known to be equal to the class of languages, definable by first order formulas using only succ but not having any restrictions on the number of variables used. At last we will show that the above class, obtained by using **X**, **Y** and **D**, is equal to the subclass of all locally threshold testable languages, obtained when allowing no higher threshold than 2.

First Order Logic	Temporal Logic	Semigroups	Combinatorics	Shown in
$\mathsf{FO}[<, \text{succ}]$	LTL	$\mathbf{A}$		[Sch65], [MP71], [Kam68]
$\mathsf{FO}^2[<, \text{succ}]$	$\mathsf{LTL}[\mathbf{F}, \mathbf{O}, \mathbf{X}, \mathbf{Y}]$	$\mathbf{DA} * \mathbf{D}$		[EVW97], [TW98]
$\mathsf{FO}^2[<]$	$\mathsf{LTL}[\mathbf{F}, \mathbf{O}]$	$\mathbf{DA}$		[EVW97], [TW98]
$\mathsf{FO}^2[\text{succ}]$	$\mathsf{LTL}[\mathbf{D}, \mathbf{X}, \mathbf{Y}]$		locally threshold testable with threshold 2	this thesis
$\mathsf{FO}[\text{succ}]$			locally threshold testable	[Str94]

List of all classes covered in this thesis

Chapter 2 contains the necessary terminology and some basic results while chapter 3 contains the main results already mentioned above and chapters 4–6 contain the proofs of these results.

2 Preliminaries

We will begin by introducing the terminology needed for the later results. Therefore we will start with an introduction into formal languages and their connection to semigroups. Then we will get to know how first order formulas can be used in the formal language context and finally define temporal logic, an extension of propositional logic which is in general not as powerful as first order logic, but we will show in the next chapter that – in the formal language context – it is equivalent to first order logic with certain predicates.

2.1 Formal Languages and Finite Semigroups

Definition 2.1. (Alphabet, Word and Language)
An *alphabet* is an arbitrary finite set whose elements are called *letters*. A *(finite) word* over such an alphabet is a finite sequence of letters. The empty word of length 0 is denoted by ϵ. A *(formal) language* is an arbitrary set of words over a fixed alphabet. If Σ is an alphabet then we write Σ^* for the language containing all words over Σ and Σ^+ for $\Sigma^* \setminus \{\epsilon\}$. Instead of $(a_0, a_1, \ldots, a_k)$ we write $a_0 a_1 \cdots a_k$ and for a word $a \in \Sigma^*$ we write a_i for the $(i+1)$th letter in a, $|a|$ for the length of a and $\lambda(a)$ for the set $\{a_0, a_1, \ldots, a_{|a|-1}\}$ of letters occuring in a.

Definition 2.2. (Syntactic Semigroup)
For an arbitrary alphabet Σ we define the *concatenation* of words $a, b \in \Sigma^*$ as $a \circ b := a_0 a_1 \cdots a_k b_0 b_1 \cdots b_l$ where $a = a_0 \cdots a_k$ and $b = b_0 \cdots b_l$. Obviously $\circ$ is an associative binary operation on Σ^* and hence $(\Sigma^*, \circ)$ as well as $(\Sigma^+, \circ)$ are semigroups. We often write ab instead of $a \circ b$. If $L \subseteq \Sigma^*$ we define the *syntactic congruence* $\approx_L$ of L as follows:

$$\text{For all } a, b \in \Sigma^*: \quad a \approx_L b \text{ iff for all } u, v \in \Sigma^*: uav \in L \Leftrightarrow ubv \in L$$

$\approx_L$ is a congruence relation on Σ^* and on Σ^+ for every language $L \subseteq \Sigma^*$ with respect to the operation of concatenation of words and therefore $\Sigma^+/_{\approx_L}$ and $\Sigma^*/_{\approx_L}$ are semigroups; the former one is called the *syntactic semigroup* of L and the latter one is called the *syntactic monoid* of L.

Usually one only considers $\Sigma^*/_{\approx_L}$ and does not also discuss $\Sigma^+/_{\approx_L}$ but here we have the problem that in the context of first order logic – which we will cover in the next section – we cannot include the empty word into any language and so we have to restrict ourselves to the analysis of $\Sigma^+/_{\approx_L}$. However, this is not really a problem as including the empty word in a given language $L \subseteq \Sigma^+$ does not change any property of L relevant for us – given that it is at all possible to include the empty word. And therefore we will from now on not mention this rather irrelevant problem anymore.

Definition 2.3. (Regular language)
A formal language $L \subseteq \Sigma^*$ is called *regular* iff its syntactic semigroup is finite.

We will now briefly leave the context of formal languages and discuss general classes of semigroups. The connection back to formal languages is of course always present in the syntactic semigroup of a language.

For the next definition we need the following remark.

Remark 2.4.
Let S be an arbitrary finite semigroup and $a \in S$. Then there is a unique $b \in \{a, a^2, a^3, \dots\}$ with b idempotent, i.e. $b^2 = b$.

Proof. Existence. Because S is finite, the set $\{a, a^2, a^3, \dots\}$ is finite as well. Therefore there are $k, l \in \mathbb{N} \setminus \{0\}$ such that $k = \min\{i > 0 \mid a^i = a^{i+j} \text{ for a } j \in \mathbb{N} \setminus \{0\}\}$ and $l = \min\{j > 0 \mid a^k = a^{k+j}\}$. By induction follows that

$$\text{for all } i, j \in \mathbb{N}, i \geq k : a^i = a^{i+jl}.$$

Now let $n \in \mathbb{N}$ such that $k \leq n < k + l$ and that there is an $m \in \mathbb{N}$ with $ml = n$. Then $a^n = a^{ml} = a^{ml+ml} = a^{2n}$.

Uniqueness. Let $p \in \mathbb{N} \setminus \{0\}$ arbitrarily chosen with $a^p = a^{2p}$. Since $a^p = a^{2p} = a^{4p} = a^{8p} = \dots$ we can w.l.o.g. assume that $p \geq k$ and $p \geq l$. Let $q, r \in \mathbb{N}$ such that $p = ql + r$ and $r < l$. Since $a^k = a^{k-r+ql+r} = a^{k-r+p} = a^{k-r+2p} = a^{k-r+2ql+2r} = a^{k+r}$ and $r < l$ it follows that $r = 0$ and therefore $p = ql$. Because $p \geq k$ we have $p \geq n$ and therefore $q \geq m$, $p = ml + (q - m)l$ and $a^p = a^{ml+(q-m)l} = a^{ml} = a^n$. □

Definition 2.5. (**A**, **D** and **DA**)
For a finite semigroup S and $a \in S$ let a^ω be the element of the set $\{a, a^2, a^3, \dots\}$ which is idempotent and let $\overline{S}$ be the monoid induced by S, i.e.

$$\overline{S} := \begin{cases} S & \text{if } S \text{ already contains an identity element} \\ S \cup \{e\} & \text{if } S \text{ does not contain an identity element} \end{cases},$$

where e is a new identity element for S.

Then the class **A** of aperiodic semigroups is defined as the class of all finite semigroups S which satisfy the condition

$$a^\omega a = a^\omega \text{ for all } a \in \overline{S}.$$

DA is the class of all finite semigroups S which satisfy the condition

$$(abc)^\omega b (abc)^\omega = (abc)^\omega \text{ for all } a, b, c \in \overline{S}.$$

And **D** is the class of all finite semigroups S which satisfy the condition

$$ba^\omega = a^\omega \text{ for all } a, b \in \overline{S}.$$

Classes of semigroups as the three defined above are often called *pseudovarieties*.

Remark 2.6.
We write a^ω because in the case that $S \in \mathbf{A}$, we get for $a \in S$ that a^ω is the element that results when multiplying a with itself infinitely many times. If $S \notin \mathbf{A}$ there is no such unique element but still a^ω occurs infinitely often when multiplying a with itself infinitely often.

Remark 2.7.
One could think that the condition defining **A** *is satisfied in every semigroup as it does not matter whether you multiply a with itself infinitely often or infinitely often and then one time more. But as stated in the above remark this view of a^ω is not always valid. Consider for example the semigroup $\mathbb{Z}_2 := (\{0,1\},+)$ where $+$ is defined by*

$+$	0	1
0	0	1
1	1	0

.

Then $1^\omega = 0$ but $1^\omega + 1 = 1$.

Remark 2.8.
DA $\subset$ **A**.

Proof. Let $S \in \mathbf{DA}$ and $b \in \overline{S}$. We have to show that $b^\omega b = b^\omega$. Since $S \in \mathbf{DA}$ we get $(ebe)^\omega b (ebe)^\omega = (ebe)^\omega$ (where e is the neutral element of $\overline{S}$) and therefore $b^\omega b b^\omega = b^\omega$. Now let $l, m \in \mathbb{N}$ be defined as in the proof of Remark 2.4, i.e. $b^\omega = b^{ml} = b^{(m+1)l} = \cdots = b^{2ml}$. Then we get $b^\omega b b^\omega = b^{ml} b b^{ml} = b^{ml+1+ml} = b^{2ml+1} = b^{ml+1} = b^\omega b$ and therefore the wanted result. □

The pseudovariety **A** plays an important role in Theorem 3.1 and **DA** plays an important role in Theorems 3.2 and 3.3 whereas **D** is only needed in the part of Theorem 3.2 which we will only quote and not prove.

2.2 First Order Logic

We will not give a general introduction into first order logic here but just describe the vocabularies and structures we will use in the later sections and chapters. The reader may refer to [EFT94] or any other introductory textbook on mathematical logic for a general introduction into first order logic.

Definition 2.9.
An $\mathsf{FO}[<, \mathrm{succ}]$-*formula* is a first order (with equality) formula over the vocabulary $\{<, \mathrm{succ}, P_0, P_1, P_2, \dots\}$, where $<, \mathrm{succ}$ are binary relation symbols and all P_i are unary relation symbols. $\mathsf{FO}[<, \mathrm{succ}]$ is the set of all $\mathsf{FO}[<, \mathrm{succ}]$-formulas without free variables (also called sentences) and $\mathsf{FO}[<]$ and $\mathsf{FO}[\mathrm{succ}]$ are the sets of all $\mathsf{FO}[<, \mathrm{succ}]$-sentences in which no succ respectively no $<$ occurs. Finally we write $\mathsf{FO}^k[<, \mathrm{succ}], \mathsf{FO}^k[<]$ and $\mathsf{FO}^k[\mathrm{succ}]$ for $\mathsf{FO}[<, \mathrm{succ}]$ respectively $\mathsf{FO}[<], \mathsf{FO}[\mathrm{succ}]$ restricted to sentences with at most k different variables – each of which may be quantified arbitrarily often in a single formula.

Because "=" is an element of our first order language, we use the symbol "≡" to denote the equality of two formulas.

Example 2.10.
The formula $\psi := \exists x\Big(\big(\forall y\ \neg y < x\big) \wedge \exists y\Big(x < y \wedge P_0(y) \wedge \exists x\big(x < y \wedge P_2(x)\big)\Big)\Big)$ *is in* $\mathsf{FO}^2[<]$.

As this example shows, the quantifier depth of a formula is not even limited when the number of different variables is. This is due to the fact that a variable can be introduced, then have another variable relate to it, e.g. by $<$, and finally be "redefined" with relation to that same other variable which is still "active".

Our goal is to associate a single formula with a whole language, i.e. a set of words. Therefore we first need to relate single words with single formulas. For that we will define the structure of a word and then define the relationship between a word and a formula as that between the structure of the word and the formula.

Definition 2.11. (Structure of a word)
For every nonempty word $a = a_0a_1 \cdots a_{n-1}$ over the alphabet $\Sigma_m := \{b_0, b_1, \ldots, b_m\}$ we define the first order structure $\overline{a}$ of a as $\overline{a} := (\{0, 1, \ldots, n-1\}, <^a, \mathrm{succ}^a, P_0^a, P_1^a, \ldots, P_m^a)$ where $\{0, 1, \ldots, n-1\}$ is the universe, $<^a$ is the usual order relation on $\{0, 1, \ldots, n-1\}$, succ^a is the successor relation on $\{0, 1, \ldots, n-1\}$, i.e. $\mathrm{succ}^a := \{(k, k+1) \mid k \in \{0, 1, \ldots, n-2\}\}$ and for all $i \leq m : P_i^a := \{j \mid a_j = b_i\}$, i.e. the set of all positions in a where the letter b_i occurs. Note that we only consider nonempty words because for the empty word the universe of the structure would be the empty set and this is generally not desirable for first order logic as all quantifiers would lose their meaning.

We usually identify $\overline{a}$ with a, e.g. we say that a *fulfills* φ (and also write $a \models \varphi$) iff $\overline{a} \models \varphi$.

From now on we will mostly speak about Σ_m for an arbitrary natural number m instead of completely arbitrary alphabets. This is only due to representational reasons (P_i instead of P_{b_i} and the like) and not a real restriction since for every alphabet Σ the letters can simply be relabeled such that Σ transforms into Σ_m for an m. Note that we do not need any order of the letters – the order induced by numbering the letters is a never used byproduct.

Definition 2.12. (Recognized language)
Let $\varphi \in \mathsf{FO}[<, \mathrm{succ}]$ and let $k := max\{i \mid P_i \text{ occurs in } \varphi\}$. Then we define the language *recognized by* φ as follows: $\mathcal{L}_\varphi := \{a \in \Sigma_k^+ \mid a \models \varphi\}$. As stated before we only consider nonempty words as the structure of the empty word is the empty set and therefore it can neither fulfill nor not fulfill any formula.

2.3 Ehrenfeucht-Fraïssé Games

We will now define a game theoretic view of first order logic which can be used to show whether some property is first order definable and will be used extensively in the later chapters as connection between first order logic on the one hand and finite semigroups and combinatorics on the other hand. A good introduction to the topic can be found in [Imm99, pp. 91-112] by Immerman whose notation we mostly adopt.

Definition 2.13. (Induced Substructure)
Let $\mathcal{A}$ be an arbitrary structure and C a subset of the universe of $\mathcal{A}$. If the vocabulary of $\mathcal{A}$ does not contain any functions or constants, we define the *induced substructure* $C^{\mathcal{A}}$ as the structure over the same vocabulary as $\mathcal{A}$ with universe C and where every relation $R^{C^{\mathcal{A}}}$ is the restriction of the original relation $R^{\mathcal{A}}$ to the new universe C.

Definition 2.14. (Isomorphism)
Let $\mathcal{A}$ and $\mathcal{B}$ be structures over the same vocabulary which does not contain any functions or constants, let A and B be their respective universes and $\varphi : A \to B$ a total function. Then φ is called an *isomorphism* iff φ is bijective and for every relation R (with arity i_R) of the common vocabulary of $\mathcal{A}$ and $\mathcal{B}$ we have

$$\text{For all } a_1, a_2, \ldots, a_{i_R} \in A: \ (a_1, a_2, \ldots, a_{i_R}) \in R^{\mathcal{A}} \ \Leftrightarrow \ (\varphi(a_1), \varphi(a_2), \ldots, \varphi(a_{i_R})) \in R^{\mathcal{B}}.$$

Definition 2.15. (Ehrenfeucht-Fraïssé Game)
Let $k, r \in \mathbb{N}$. An *r-round k-pebble Ehrenfeucht-Fraïssé game* is played on two structures $\mathcal{A}$ and $\mathcal{B}$ over the same vocabulary. It is played by a male player named Spoiler and a female player named Duplicator over a maximum of r rounds. There are k pairwise distinct pebbles for each structure which are all lying outside the structures at the beginning of the game. In each round Spoiler chooses one of the two structures and one of the k pebbles belonging to this structure and puts the pebble on an arbitrary element of the universe of the structure. He may choose a pebble already lying on an element of the universe – in this case he simply moves the pebble. Duplicator must then respond by putting the corresponding pebble belonging to the other structure on an element of the universe of this other structure.

If the structures are words – in the sense of definition 2.11 – we denote the position of the l-th pebble on the first (respectively second) word after round s is played by $\alpha_s(x_l)$ (respectively $\beta_s(x_l)$). So $\alpha_s : \{x_1, \ldots, x_k\} \to \{0, \ldots, |a| - 1\}$ and $\beta_s : \{x_1, \ldots, x_k\} \to \{0, \ldots, |b| - 1\}$ are partial functions for every $s \in \{0, \ldots, r\}$.

Duplicator wins round s iff $\beta_s \circ \alpha_s^{-1}$ is an isomorphism of the induced substructures $\text{range}(\alpha)^{\mathcal{A}}$ and $\text{range}(\beta)^{\mathcal{B}}$. Duplicator wins the game iff she wins all rounds $s \in \{1, \ldots, r\}$.

Note that α_s^{-1} is not a function if $\alpha_s(x_i) = \alpha_s(x_j)$ for some $s, i, j \in \mathbb{N}$. But if also $\beta_s(x_i) = \beta_s(x_j)$ it can still be true that $\beta_s \circ \alpha_s^{-1}$ is a bijective function.

Since Ehrenfeucht-Fraïssé games are deterministic games of perfect information and finite length there is always a *winning strategy* for one of the two players, i.e. one of the two players can play in such a way that his or her opponent can do nothing against losing the game.

Definition 2.16.
Let $m, k, r \in \mathbb{N}$ and $a, b \in \Sigma_m^+$. Then $a \cong_r^k b$ iff Duplicator has a winning strategy in the r-round k-pebble Ehrenfeucht-Fraïssé game on a and b.

For this definition the underlying first order vocabulary is important because whether a function is an isomorphism or not depends on which relations are considered. The

underlying vocabulary would naturally be $\{<, \text{succ}, P_0, P_1, P_2, \dots\}$ for we defined the structure of a word as a structure over this vocabulary; but when we deal with fragments of $\mathsf{FO}[<, \text{succ}]$ we will also consider only subsets of the above vocabulary. Specifically when we cover fragments of $\mathsf{FO}[<]$ we will take $\{<, P_0, P_1, P_2, \dots\}$ as the underlying vocabulary and when we deal with fragments of $\mathsf{FO}[\text{succ}]$ we will take $\{\text{succ}, P_0, P_1, P_2, \dots\}$ as the underlying vocabulary. Usually it will be clear from the context which vocabulary is considered.

It can be easily seen that $\cong_r^k$ is a congruence relation for every $k, r \in \mathbb{N}$ with respect to the operation of concatenation of words. When we write about congruence relations we will from now on always mean "with respect to the operation of concatenation of words".

2.4 Temporal Logic

Temporal logic is an extension of propositional logic which can very well be used to describe properties of computing systems – and was partly developed for that purpose. However, we will use it to recognize formal languages – in the same sense as for first order logic.

In the next chapters it will be our main goal to provide different characterizations of several classes of formal languages corresponding to certain classes of temporal logic formulas.

Definition 2.17. (Syntax)
The set LTL of all *(linear) temporal logic formulas* is inductively defined as follows:

- For all $i \in \mathbb{N}$: $p_i \in \mathsf{LTL}$
- For all $\varphi, \psi \in \mathsf{LTL}$: $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$, $\varphi \leftrightarrow \psi$, $\mathbf{X}\varphi$, $\mathbf{Y}\varphi$, $\mathbf{F}\varphi$, $\mathbf{O}\varphi$, $\mathbf{D}\varphi$, $\varphi\mathbf{XU}\psi$, $\varphi\mathbf{YS}\psi \in \mathsf{LTL}$

$p_0, p_1, p_2, \dots$ are called *atomic propositions* and $\mathbf{X}$(ne$\mathbf{X}$t), $\mathbf{Y}$(esterday), $\mathbf{F}$(uture), $\mathbf{O}$(nce), $\mathbf{D}$(ifferent), $\mathbf{XU}$(ne$\mathbf{X}$t $\mathbf{U}$ntil) and $\mathbf{YS}$($\mathbf{Y}$esterday $\mathbf{S}$ince) are called temporal operators. For any temporal operators $Op1, Op2, \dots$ we write $\mathsf{LTL}[Op1, Op2, \dots]$ for the subset of LTL which uses no temporal operators except $Op1, Op2, \dots$. $\mathsf{LTL}[\mathbf{X}, \mathbf{Y}, \mathbf{F}, \mathbf{O}, \mathbf{D}]$ is called *unary temporal logic* (UTL).

Example 2.18.
The formula $\varphi := \mathbf{F}(p_0 \wedge \mathbf{O}p_2)$ *is in* $\mathsf{LTL}[\mathbf{F}, \mathbf{O}]$.

The semantics of temporal logic formulas are defined by a translation to $\mathsf{FO}[<, \text{succ}]$-sentences.

Definition 2.19. (Semantics)
Let $\varphi \in \mathsf{LTL}$. Then we inductively define a corresponding $\mathsf{FO}[<, \text{succ}]$-formula $\hat{\varphi}(x)$ as follows:

- If $\varphi = p_i$ for some $i \in \mathbb{N}$ then $\hat{\varphi}(x) := P_i(x)$.
- If $\varphi = \psi \wedge \theta$ then $\hat{\varphi}(x) := \hat{\psi}(x) \wedge \hat{\theta}(x)$. Analogous for $\neg$, $\vee$, $\rightarrow$ and $\leftrightarrow$.
- If $\varphi = \mathbf{X}\psi$ then $\hat{\varphi}(x) := \exists y(\mathrm{succ}(x, y) \wedge \hat{\psi}(y))$.
- If $\varphi = \mathbf{Y}\psi$ then $\hat{\varphi}(x) := \exists y(\mathrm{succ}(y, x) \wedge \hat{\psi}(y))$.
- If $\varphi = \mathbf{F}\psi$ then $\hat{\varphi}(x) := \exists y(x < y \wedge \hat{\psi}(y))$.
- If $\varphi = \mathbf{O}\psi$ then $\hat{\varphi}(x) := \exists y(y < x \wedge \hat{\psi}(y))$.
- If $\varphi = \mathbf{D}\psi$ then $\hat{\varphi}(x) := \exists y(\neg\, x = y \wedge \neg\, \mathrm{succ}(x, y) \wedge \neg\, \mathrm{succ}(y, x) \wedge \hat{\psi}(y))$.
- If $\varphi = \psi\mathbf{XU}\theta$ then $\hat{\varphi}(x) := \exists y\Big(x < y \wedge \hat{\theta}(y) \wedge \forall z\big(x < z \wedge z < y \;\rightarrow\; \hat{\psi}(z)\big)\Big)$.
- If $\varphi = \psi\mathbf{YS}\theta$ then $\hat{\varphi}(x) := \exists y\Big(y < x \wedge \hat{\theta}(y) \wedge \forall z\big(z < x \wedge y < z \;\rightarrow\; \hat{\psi}(z)\big)\Big)$.

Finally we define $\overline{\varphi} := \exists x\Big(\big(\forall y\, \neg\, y < x\big) \wedge \hat{\varphi}(x)\Big) \in \mathsf{FO}[<, \mathrm{succ}]$ and define the semantics of φ as those of $\overline{\varphi}$, i.e. $a \models \varphi$ iff $a \models \overline{\varphi}$ and $\mathcal{L}_\varphi := \mathcal{L}_{\overline{\varphi}}$. We also write $a, i \models \varphi$ for $a \models \hat{\varphi}(i)$.

Note that $a \models \overline{\varphi}$ iff $a \models \hat{\varphi}(0)$ and therefore $a \models \varphi$ can simply be viewed as an abbreviation for $a, 0 \models \varphi$.

Intuitively, $a, i \models \varphi$ stands for "the word a at the position i fulfills the formula φ".

Example 2.20.
The formulas ψ from Example 2.10 and φ from Example 2.18 have the following connection:

$$\psi \equiv \overline{\varphi}$$

Definition 2.21.
Two formulas φ, ψ (each in LTL or FO) are called *equivalent* iff $\mathcal{L}_\varphi = \mathcal{L}_\psi$. If $\varphi, \psi \in \mathsf{LTL}$ they are called *strongly equivalent* iff for all words a and all $i \in \{0, \ldots, |a| - 1\}$: $a, i \models \varphi$ iff $a, i \models \psi$.

Our major results will only use the notion of (not necessarily strong) *equivalence* as we are mainly interested in recognizable languages. However, when dealing with formulas which are to be used as subformulas of other formulas it can be of importance whether the formulas are only equivalent "at the beginning" of a word or also "at every position inside" the word; and this exactly is the difference between *strong equivalence* and *equivalence*.

Remark 2.22.
We can translate every LTL*-formula into a strongly equivalent* $\mathsf{LTL}[\mathbf{XU}, \mathbf{YS}]$*-formula according to the following rules:*

- $\mathbf{X}\varphi$ *translates into* $(p_0 \wedge \neg p_0)\mathbf{XU}\varphi$.

- $\mathbf{Y}\varphi$ *translates into* $(p_0 \wedge \neg p_0)\mathbf{YS}\varphi$.
- $\mathbf{F}\varphi$ *translates into* $(p_0 \vee \neg p_0)\mathbf{XU}\varphi$.
- $\mathbf{O}\varphi$ *translates into* $(p_0 \vee \neg p_0)\mathbf{YS}\varphi$.
- $\mathbf{D}\varphi$ *translates into* $\mathbf{XF}\varphi \vee \mathbf{YO}\varphi$.

We only have defined the semantics of the other formulas by a translation to first order logic because of the next Remark and Theorems 3.2, 3.3 and 3.4.

Remark 2.23.
The following connections between subclasses of LTL *and subclasses of* $\mathsf{FO}[<, \mathrm{succ}]$ *hold true:*

- *If* $\varphi \in \mathsf{LTL}[\mathbf{X}, \mathbf{Y}, \mathbf{D}]$ *then* $\overline{\varphi} \in \mathsf{FO}^2[\mathrm{succ}]$.
  (We have to change the old definition of $\overline{\varphi}$ *to the equivalent new definition* $\overline{\varphi} := \exists x\Big(\big(\forall y\ \neg\,\mathrm{succ}(y,x)\big) \wedge \hat{\varphi}(x)\Big)$ *for this to be true.)*
- *If* $\varphi \in \mathsf{LTL}[\mathbf{F}, \mathbf{O}]$ *then* $\overline{\varphi} \in \mathsf{FO}^2[<]$.
- *If* $\varphi \in \mathsf{UTL}$ *then* $\overline{\varphi} \in \mathsf{FO}^2[<, \mathrm{succ}]$.
- *If* $\varphi \in \mathsf{LTL}[\mathbf{XU}, \mathbf{YS}]$ *then* $\overline{\varphi} \in \mathsf{FO}^3[<]$.
- *If* $\varphi \in \mathsf{LTL}$ *then* $\overline{\varphi}$ *is equivalent to a formula in* $\mathsf{FO}^3[<]$.

Remark 2.24.
The operator $\mathbf{D}$ *is not one of the commonly used temporal operators but is our "invention". One would expect* $\mathbf{D}\varphi$ *to be equivalent to* $\mathbf{F}\varphi \vee \mathbf{O}\varphi$ *but instead we have the connection from 2.22 which suggests to call the* $\mathbf{D}$ *operator "very different" rather than "different". However, for the context of Remark 2.23 it does not matter because for the operator* $\mathbf{D}'$ *defined by* $\mathbf{D}'\varphi := \mathbf{F}\varphi \vee \mathbf{O}\varphi$ *it follows that:*

For all $\varphi \in \mathsf{LTL}[\mathbf{X}, \mathbf{Y}, \mathbf{D}]$ *there is a* $\varphi' \in \mathsf{LTL}[\mathbf{X}, \mathbf{Y}, \mathbf{D}']$ *strongly equivalent to* φ
and for all $\psi \in \mathsf{LTL}[\mathbf{X}, \mathbf{Y}, \mathbf{D}']$ *there is a* $\psi' \in \mathsf{LTL}[\mathbf{X}, \mathbf{Y}, \mathbf{D}]$ *strongly equivalent to* ψ.

Proof. Both statements will be proven inductively over the structure of the formula. For both statements it is clear that Boolean connections as well as $\mathbf{X}$ and $\mathbf{Y}$ translate one to one. So we only need to translate formulas of the form $\mathbf{D}\varphi$ respectively $\mathbf{D}'\psi$.

The second translation is the easy one. So let $\mathbf{D}'\psi \in \mathsf{LTL}[\mathbf{X}, \mathbf{Y}, \mathbf{D}']$ and assume by induction hypothesis that ψ is equivalent to a formula $\psi' \in \mathsf{LTL}[\mathbf{X}, \mathbf{Y}, \mathbf{D}]$. Then it follows that $\mathbf{D}'\psi$ is equivalent to $\mathbf{F}\psi' \vee \mathbf{O}\psi'$ which is equivalent to

$$\mathbf{X}\psi' \vee \mathbf{XF}\psi' \ \vee\ \mathbf{Y}\psi' \vee \mathbf{YO}\psi'.$$

And this is equivalent to

$$\mathbf{D}\psi' \vee \mathbf{X}\psi' \vee \mathbf{Y}\psi' \ \in \mathsf{LTL}[\mathbf{X}, \mathbf{Y}, \mathbf{D}]$$

by Remark 2.22.

The first translation is not really more difficult but rather more complicated. In the following translation we use the notation $[\psi_1, \psi_2, \psi_3]$ for the formula $\mathbf{Y}\psi_1 \wedge \psi_2 \wedge \mathbf{X}\psi_3$. Now let $\mathbf{D}\varphi \in \mathsf{LTL}[\mathbf{X}, \mathbf{Y}, \mathbf{D}]$ and assume by induction hypothesis that φ is equivalent to a formula $\varphi' \in \mathsf{LTL}[\mathbf{X}, \mathbf{Y}, \mathbf{D}']$. Then it follows that $\mathbf{D}\varphi$ is equivalent to

$$
\begin{array}{lll}
 & \Big([\neg\varphi', \neg\varphi', \neg\varphi'] & \wedge \quad \mathbf{D}'\varphi' \Big) \\
\vee & \Big([\neg\varphi', \varphi', \neg\varphi'] & \wedge \quad \mathbf{D}'\varphi' \Big) \\
\vee & \Big([\varphi', \neg\varphi', \neg\varphi'] & \wedge \quad \mathbf{Y}\mathbf{D}'\varphi' \Big) \\
\vee & \Big([\neg\varphi', \neg\varphi', \varphi'] & \wedge \quad \mathbf{X}\mathbf{D}'\varphi' \Big) \\
\vee & \Big([\varphi', \neg\varphi', \varphi'] & \wedge \quad \Big(\mathbf{D}'\big(\varphi' \wedge \neg(\mathbf{X}\neg\varphi' \wedge \mathbf{X}\mathbf{X}\varphi') \wedge \neg(\mathbf{Y}\neg\varphi' \wedge \mathbf{Y}\mathbf{Y}\varphi')\big) \vee \mathbf{D}'[\varphi', \neg\varphi', \varphi'] \Big) \Big) \\
\vee & \Big([\neg\varphi', \varphi', \varphi'] & \wedge \quad \Big(\mathbf{D}'\big(\varphi' \wedge \neg(\mathbf{Y}\varphi' \wedge \mathbf{Y}\mathbf{Y}\neg\varphi')\big) \vee \mathbf{D}'[\neg\varphi', \varphi', \varphi'] \Big) \Big) \\
\vee & \Big([\varphi', \varphi', \neg\varphi'] & \wedge \quad \Big(\mathbf{D}'\big(\varphi' \wedge \neg(\mathbf{X}\varphi' \wedge \mathbf{X}\mathbf{X}\neg\varphi')\big) \vee \mathbf{D}'[\varphi', \varphi', \neg\varphi'] \Big) \Big) \\
\vee & \Big([\varphi', \varphi', \varphi'] & \wedge \quad \Big(\mathbf{D}'\big(\varphi' \wedge \neg(\mathbf{X}\varphi' \wedge \mathbf{X}\mathbf{X}\varphi') \wedge \neg(\mathbf{Y}\varphi' \wedge \mathbf{Y}\mathbf{Y}\varphi')\big) \vee \mathbf{D}'[\varphi', \varphi', \varphi'] \Big) \Big) \\
\vee & \text{“several simple cases for the beginning and end of a word”}
\end{array}
$$

The problem is that $\mathbf{D}'$ can only exclude one position at a time and so something like $\mathbf{D}'\varphi' \wedge \mathbf{X}\mathbf{D}'\varphi'$ would be satisfied by any model of $\varphi' \wedge \mathbf{X}\varphi'$ which obviously does not need to satisfy $\mathbf{D}\varphi$. To cope with this we make a case distinction for the validity of φ' on the three positions which we do not want to say anything about. The knack is to express – in the difficult cases where φ' holds at least two times in the neighbourhood of the current position – that there either must be a position where φ' holds and which has a different neighbourhood or there must be a second position with the exact same neighbourhood. This way it is guaranteed that there is a position not in the neighbourhood of the current position such that φ' holds there. □

The general idea of the last proof – to do a case distinction and then reveal and use the structure of the problem – will occur again in the proof of Theorem 6.5.

3 Main Results

This chapter contains the main theorems of this thesis but their detailed proofs make up the next three chapters.

We start with a theorem about different characterizations of $\mathsf{FO}[<, \mathrm{succ}]$, i.e. first order logic with $<$ and succ and without any restriction on the number of variables used in a formula. However, we just mention it for the sake of completeness and will not prove it here. We then move on to our real goal: Different characterizations of $\mathsf{FO}^2[<, \mathrm{succ}]$, $\mathsf{FO}^2[<]$ and $\mathsf{FO}^2[\mathrm{succ}]$, i.e. first order logic restricted to formulas with at most two different variables and sometimes with only one of our two binary predicates. Theorems 3.2 and 3.3 are old results – although not nearly as old as Theorem 3.1 – but Theorem 3.4 is new in this thesis.

From now on always let m be an arbitrary natural number.

Theorem 3.1.
Let $L \subseteq \Sigma_m^+$. Then the following are equivalent:

(i) *The syntactic semigroup of L is in $\mathbf{A}$.*

(ii) *$L = \mathcal{L}_\varphi$ for a $\varphi \in \mathsf{FO}[<, \mathrm{succ}]$.*

(iii) *$L = \mathcal{L}_\varphi$ for a $\varphi \in \mathsf{LTL}$.*

(iv) *$L = \mathcal{L}_\varphi$ for a $\varphi \in \mathsf{LTL}[\mathbf{XU}, \mathbf{YS}]$.*

(v) *$L = \mathcal{L}_\varphi$ for a $\varphi \in \mathsf{FO}^3[<]$.*

This theorem goes back to [Sch65], [MP71] ($i \Leftrightarrow ii$) and [Kam68] ($ii \Rightarrow iii$). We just notice that the implication from *(iii)* to *(iv)* is Remark 2.22, the implication from *(iv)* to *(v)* is contained in Remark 2.23 and for the implication from (v) to (ii) there is nothing to show. The interested reader may refer to Diekert and Gastin [DG08] for a complete proof (and some more equivalences).

Theorem 3.2.
Let $L \subseteq \Sigma_m^+$. Then the following are equivalent:

(i) *The syntactic semigroup of L is in $\mathbf{DA} * \mathbf{D}$, the* semidirect product *of $\mathbf{DA}$ and $\mathbf{D}$.*

(ii) *$L = \mathcal{L}_\varphi$ for a $\varphi \in \mathsf{FO}^2[<, \mathrm{succ}]$.*

(iii) *$L = \mathcal{L}_\varphi$ for a $\varphi \in \mathsf{UTL}$.*

Proof. *(i)* $\Leftrightarrow$ *(ii)*: This follows from Theorem 5.3 by the wreath product principle from Straubing [Str78] and was shown in [TW98]. We will not give the proof here, nor say what the wreath product principle is, as this would require too much semigroup theory. We will not even define the semidirect product as we will never need it again. So this part of the theorem exists only for those experienced in semigroup theory.

(ii) $\Rightarrow$ *(iii)*: This is Proposition 4.1 a) which was first proved in [EVW97].

(iii) $\Rightarrow$ *(ii)*: This is contained in Remark 2.23.

□

Theorem 3.3.
Let $L \subseteq \Sigma_m^+$. Then the following are equivalent:

(i) The syntactic semigroup of L is in $\mathbf{DA}$.

(ii) $L = \mathcal{L}_\varphi$ *for a* $\varphi \in \mathsf{FO}^2[<]$.

(iii) $L = \mathcal{L}_\varphi$ *for a* $\varphi \in \mathsf{LTL}[\mathbf{F}, \mathbf{O}]$.

Proof. *(i)* $\Leftrightarrow$ *(ii)*: This is contained in Theorem 5.3 which was first proved in [TW98].

(ii) $\Rightarrow$ *(iii)*: This is Proposition 4.1 b) which was first proved in [EVW97].

(iii) $\Rightarrow$ *(ii)*: This is contained in Remark 2.23.

□

Theorem 3.4.
Let $L \subseteq \Sigma_m^+$. Then the following are equivalent:

(i) $L = \mathcal{L}_\varphi$ *for a* $\varphi \in \mathsf{FO}^2[\mathrm{succ}]$.

(ii) $L = \mathcal{L}_\varphi$ *for a* $\varphi \in \mathsf{LTL}[\mathbf{X}, \mathbf{Y}, \mathbf{D}]$.

(iii) L is locally threshold testable with threshold 2.

Proof. *(i)* $\Rightarrow$ *(ii)*: This is Proposition 4.1 c).

(ii) $\Rightarrow$ *(i)*: This is contained in Remark 2.23.

(i) $\Leftrightarrow$ *(iii)*: Local threshold testability is introduced in Definition 6.3 and this equivalence is Theorem 6.5.

□

Remark 3.5.
The language $\mathcal{L}_\varphi$ for $\varphi := \exists x\, P_1(x)$, i.e. $\varphi \in \mathsf{FO}^2[]$, is not definable by a formula $\psi \in \mathsf{LTL}[\mathbf{X}, \mathbf{Y}]$.

4 First Order Logic to Temporal Logic

Out of all connections stated in the previous chapter the connection between first order logic and temporal logic is probably the most basic one. This is not surprising since both are logics and temporal logic is even defined by a translation to first order logic. However, the reverse direction is not trivial but involves some work which was first done in [EVW97].

Proposition 4.1.

a) Let $\varphi \in \mathsf{FO}^2[<, \mathrm{succ}]$. *Then there is a* $\widetilde{\varphi} \in \mathsf{UTL}$ *with* $\mathcal{L}_{\widetilde{\varphi}} = \mathcal{L}_\varphi$.

b) Let $\varphi \in \mathsf{FO}^2[<]$. *Then there is a* $\widetilde{\varphi} \in \mathsf{LTL}[\mathbf{F}, \mathbf{O}]$ *with* $\mathcal{L}_{\widetilde{\varphi}} = \mathcal{L}_\varphi$.

c) Let $\varphi \in \mathsf{FO}^2[\mathrm{succ}]$. *Then there is a* $\widetilde{\varphi} \in \mathsf{LTL}[\mathbf{X}, \mathbf{Y}, \mathbf{D}]$ *with* $\mathcal{L}_{\widetilde{\varphi}} = \mathcal{L}_\varphi$.

Proof. This proof is basically from [EVW97]. It is slightly modified to include cases b) and c) and, of course, it is extended to (hopefully) improve clarity.

The proofs of a), b) and c) are almost identical. So we will give the proof for a) and point out the changes needed for b) and c) where they are necessary.

W.l.o.g. we can assume the only variables in φ are x and y. If φ is a Boolean connection of subformulas, i.e. $\varphi \equiv \psi \wedge \theta$ or $\varphi \equiv \neg\psi$ for $\psi, \theta \in \mathsf{FO}^2[<, \mathrm{succ}]$, we can obviously set $\widetilde{\varphi} := \widetilde{\psi} \wedge \widetilde{\theta}$ or $\widetilde{\varphi} := \neg\widetilde{\psi}$ respectively. Since φ has no free variables it cannot be an atomic formula and so the only remaining case is $\varphi \equiv \exists x\ \psi(x)$.

We will compute a formula $\psi' \in \mathsf{UTL}$ with $a \models \psi(i)$ iff $a, i \models \psi'$ for every i and every word a and set

$$\widetilde{\varphi} := \begin{cases} \psi' \vee \mathbf{F}\psi' & \text{for a), b)} \\ \psi' \vee \mathbf{X}\psi' \vee \mathbf{D}\psi' & \text{for c)} \end{cases}$$

If $\psi(x)$ is a Boolean connection of subformulas, i.e.

$$\psi(x) \equiv \theta(x) \wedge \tau(x) \text{ or } \psi(x) \equiv \neg\theta(x)$$

we construct ψ' by

$$\psi' := \theta' \wedge \tau' \text{ or } \psi' := \neg\theta'.$$

If $\psi(x) \equiv P_j x$ we set $\psi' := p_j$.

The remaining case is $\psi(x) \equiv \exists y\ \theta(x, y)$. In this case we write $\theta(x, y)$ as

$$\sigma\big(\chi_0(x, y), \ldots, \chi_{r-1}(x, y), \xi_0(x), \ldots, \xi_{s-1}(x), \zeta_0(y), \ldots, \zeta_{t-1}(y)\big)$$

where σ is a Boolean formula, each χ_i is an atomic formula (because every formula with two distinct free variables which do not get quantified again – then we would have a ξ_i

or ζ_i instead – is atomic since we are in FO^2) and each ξ_i and ζ_i is a formula with a quantifier depth less than that of ψ. We will now use the lower quantifier depth and recursively compute each ξ_i' and ζ_i', i.e. formulas ξ_i' and ζ_i' with $a, j \models \xi_i'$ iff $a \models \xi_i(j)$ and $a, j \models \zeta_i'$ iff $a \models \zeta_i(j)$ for every i, j and every word a, and then create ψ' in three steps.

Step 1. Each $\xi_i(x)$ can either be true or false and since none of them depends on y we can introduce a case distinction on all possible 2^s truth assignments for the set of all $\xi_i(x)$. We obtain the following formula which is equivalent to $\psi(x)$:

$$\bigvee_{\vec{\alpha}\in\{\top,\bot\}^s} \Big(\bigwedge_{i=0}^{s-1} \xi_i^{\alpha_i}(x) \quad \wedge \exists y\, \sigma\big(\chi_0(x,y), \ldots, \chi_{r-1}(x,y), \alpha_0, \ldots, \alpha_{s-1}, \zeta_0(y), \ldots, \zeta_{t-1}(y)\big)\Big),$$

where $\top$ stands for an arbitrary tautological formula, $\bot$ for an arbitrary unsatisfiable formula and $\xi_i^{\alpha_i}$ stands for ξ_i iff $\alpha_i \equiv \top$ and for $\neg\xi_i$ iff $\alpha_i \equiv \bot$.
The $\xi_i^{\alpha_i}(x)$ ensure that only the element of the disjunction, where all α_i get the "correct" interpretation, can be true and is therefore the only element that really counts into the disjunction.

Step 2. We will now introduce a case distinction on which of the $\chi_i(x,y)$ hold. Therefore we notice that any $\chi_i(x,y)$ only expresses some kind of order between x and y – since the only possible atomic formulas using both x and y are $x = y$, $\mathrm{succ}(x,y)$, $\mathrm{succ}(y,x)$, $x < y$ and $y < x$ (and both variables must be used because otherwise we would not really have a χ_i but a ξ_i or a ζ_i). There are only few combinations of these five formulas which can be true together and this leads us to the following set of mutually exclusive cases:

$$\Delta(x,y) := \begin{cases} \{x = y,\ \mathrm{succ}(x,y),\ \mathrm{succ}(y,x),\ x < y \wedge \neg\mathrm{succ}(x,y),\ y < x \wedge \neg\mathrm{succ}(y,x)\} & \text{for a).} \\ \{x = y,\ x < y,\ y < x\} & \text{for b).} \\ \{x = y,\ \mathrm{succ}(x,y),\ \mathrm{succ}(y,x),\ \neg x = y \wedge \neg\mathrm{succ}(x,y) \wedge \neg\mathrm{succ}(y,x)\} & \text{for c).} \end{cases}$$

If we assume an arbitrary $\delta(x,y) \in \Delta(x,y)$ then each of the atomic formulas $\chi_i(x,y)$ will evaluate to either true or false. We will denote the value of $\chi_i(x,y)$ under $\delta(x,y)$ by χ_i^δ. We now obtain the following formula equivalent to $\psi(x)$:

$$\bigvee_{\vec{\alpha}\in\{\top,\bot\}^s} \Big(\bigwedge_{i=0}^{s-1} \xi_i^{\alpha_i}(x) \ \wedge \bigvee_{\delta\in\Delta} \exists y \big(\delta(x,y) \ \wedge \ \sigma(\chi_0^\delta, \ldots, \chi_{r-1}^\delta, \alpha_0, \ldots, \alpha_{s-1}, \zeta_0(y), \ldots, \zeta_{t-1}(y))\big)\Big).$$

Step 3. If $\delta(x,y) \in \Delta(x,y)$ and $\tau(x)$ is an $\mathsf{FO}^2[<,\mathrm{succ}]$-formula for which we already have computed $\tau' \in \mathsf{UTL}$ then we can compute τ_δ' for $\tau_\delta(x) := \exists y\ (\delta(x,y) \wedge \tau(y))$ as follows:

- If $\delta(x,y) \equiv x = y$ then we set $\tau_\delta' := \tau'$.
- If $\delta(x,y) \equiv \mathrm{succ}(x,y)$ then we set $\tau_\delta' := \mathbf{X}\tau'$.
- If $\delta(x,y) \equiv \mathrm{succ}(y,x)$ then we set $\tau_\delta' := \mathbf{Y}\tau'$.

- If $\delta(x,y) \equiv x < y \wedge \neg\mathrm{succ}(x,y)$ then we set $\tau'_\delta := \mathbf{XF}\tau'$.
- If $\delta(x,y) \equiv y < x \wedge \neg\mathrm{succ}(y,x)$ then we set $\tau'_\delta := \mathbf{YO}\tau'$.
- If $\delta(x,y) \equiv x < y$ then we set $\tau'_\delta := \mathbf{F}\tau'$.
- If $\delta(x,y) \equiv y < x$ then we set $\tau'_\delta := \mathbf{O}\tau'$.
- If $\delta(x,y) \equiv \neg x = y \wedge \neg\mathrm{succ}(x,y) \wedge \neg\mathrm{succ}(y,x)$ then we set $\tau'_\delta := \mathbf{D}\tau'$.

We now obtain ψ' as follows:

$$\psi' \quad = \quad \bigvee_{\overrightarrow{\alpha}\in\{\top,\bot\}^s} \Big(\bigwedge_{i=0}^{s-1} (\xi_i^{\alpha_i})' \ \wedge \ \bigvee_{\delta\in\Delta} \rho(\overrightarrow{\alpha},\delta)'_\delta \Big),$$

where $\rho(\overrightarrow{\alpha},\delta)$ is defined by

$$\rho(\overrightarrow{\alpha},\delta)(y) := \sigma(\chi_0^\delta,\ldots,\chi_{r-1}^\delta,\alpha_0,\ldots,\alpha_{s-1},\zeta_0(y),\ldots,\zeta_{t-1}(y))$$

and $\rho(\overrightarrow{\alpha},\delta)'_\delta$ can be computed by the above and by

$$\rho(\overrightarrow{\alpha},\delta)' := \sigma(\chi_0^\delta,\ldots,\chi_{r-1}^\delta,\alpha_0,\ldots,\alpha_{s-1},\zeta'_0,\ldots,\zeta'_{t-1}).$$

By induction hypothesis every $(\xi_i^{\alpha_i})' := \xi_i'^{\alpha_i}$ and every ζ'_i can be computed as each one's quantifier depth is less than that of ψ. □

5 First Order Logic and Finite Semigroups

Almost the whole chapter goes back to Thérien and Wilke [TW98] (the only exception being section 5.1). The definitions and proofs in this chapter (with the same exception as above) are also from [TW98] and are only extended and sometimes illustrated for clarity.

As already stated above the proof of the connection between first order logic and finite semigroup theory is more involved than for the inter-logic connection. In fact, it will not be proven directly but we need two intermediate characterizations. The first one is based on Ehrenfeucht-Fraïssé games and is a common approach to work with first order logic characterizations in a game-theoretic way. The second characterization was especially invented for the purpose of this proof. It is a combinatorial characterization based on certain decompositions of words. We start by defining these decompositions and a congruence relation based on them.

Definition 5.1.
Let $a \in \Sigma_m^+$ and $p \in \lambda(a)$. Then the *p-left decomposition* of a is defined as the unique triple (a', p, a'') with $a'pa'' = a$ and $p \notin \lambda(a')$ and is denoted by p-ldecomp(a). Symmetrically we define the *p-right decomposition* of a as the unique triple (a_0, p, a_1) with $a_0pa_1 = a$ and $p \notin \lambda(a_1)$ and denote it by p-rdecomp(a).

We now inductively define the equivalence relation $\equiv_n$ for every $n \in \mathbb{N}$.

Definition 5.2.
Let $a, b \in \Sigma_m^+$ and $n \in \mathbb{N}$. Then $a \equiv_n b$ iff

$$
\begin{array}{lll}
& n = 0 & \\
\text{or} & a = b & \\
\text{or} & \lambda(a) = \lambda(b) & \\
& \text{and for all } p \in \lambda(a): & (a', p, a'') = p\text{-ldecomp}(a) \text{ and } (b', p, b'') = p\text{-ldecomp}(b) \\
& & \Rightarrow\ a' \equiv_n b' \text{ and } a'' \equiv_{n-1} b'' \\
& \text{and for all } p \in \lambda(a): & (a_0, p, a_1) = p\text{-rdecomp}(a) \text{ and } (b_0, p, b_1) = p\text{-rdecomp}(b) \\
& & \Rightarrow\ a_0 \equiv_{n-1} b_0 \text{ and } a_1 \equiv_n b_1
\end{array}
$$

It can be shown that $\equiv_n$ is a congruence relation for every $n \in \mathbb{N}$.

Theorem 5.3.
Let $L \subseteq \Sigma_m^+$. Then the following are equivalent:

(i) The syntactic semigroup of L is in **DA**.

(ii) L is a union of classes of $\equiv_n$ for an $n \in \mathbb{N}$.

(iii) L *is a union of classes of* $\cong_r^2$ *(with underlying first order vocabulary* $\{<, P_0, P_1, \dots\}$*) for an* $r \in \mathbb{N}$.

(iv) $L = \mathcal{L}_\varphi$ *for a* $\varphi \in \mathsf{FO}^2[<]$.

Proof. *(i)* $\Rightarrow$ *(ii)*: This is Proposition 5.13 which was proved in [TW98].

(ii) $\Rightarrow$ *(iii)*: This is Lemma 5.5 which was proved in [TW98].

(iii) $\Rightarrow$ *(i)*: This is Lemma 5.8 which was proved in [TW98].

(iii) $\Leftrightarrow$ *(iv)*: This is Lemma 5.4 (with symbols $= \{<\}$).

□

5.1 First Order Logic and Ehrenfeucht-Fraïssé Games

This game-theoretic approach to first order logic goes back to Fraïssé [Fra54] and Ehrenfeucht [Ehr61] and has become a standard method. We will not give a complete proof for our lemma here but only cover the things specific to our formal language context and therefor use a theorem from [Imm99] about general Ehrenfeucht-Fraïssé games.

Lemma 5.4.
Let $r \in \mathbb{N}$, $L \subseteq \Sigma_m^+$ *and* symbols $\subseteq \{<, \mathrm{succ}\}$. *Then* L *is a union of classes of* $\cong_r^2$ *(with underlying first order vocabulary* symbols $\cup \{P_0, P_1, \dots\}$*) iff* $L = \mathcal{L}_\varphi$ *for a formula* $\varphi \in \mathsf{FO}^2[\text{symbols}]$ *with quantifier depth at most* r.

Proof. We use (without proof) Theorem 6.10 in [Imm99, p. 95] which states that for all $k, r \in \mathbb{N} \setminus \{0\}$ two words $a, b \in \Sigma_m^+$ are distinguishable by $\cong_r^k$, i.e. $a \ncong_r^k b$, iff they are distinguishable by a formula $\varphi \in \mathsf{FO}^k$ with quantifier depth at most r, i.e. one of the words fulfills φ and the other does not.

"$\Leftarrow$": Let $L = \mathcal{L}_\varphi$ for a $\varphi \in \mathsf{FO}^2[\text{symbols}]$ and let $r \in \mathbb{N} \setminus \{0\}$ be the quantifier depth of φ. We have to show that for every $a, b \in \Sigma_m^+$ with $a \in L$, i.e. $a \models \varphi$, and $a \cong_r^2 b$ it follows that also $b \in L$. So assume $b \notin L = \mathcal{L}_\varphi$. This yields $b \not\models \varphi$ and so a and b do not agree on φ but by the above quoted theorem this is a contradiction to our assumption that $a \cong_r^2 b$ since the quantifier depth of φ is at most r.

"$\Rightarrow$": First of all note that there are only finitely many pairwise non-equivalent formulas (two formulas are non-equivalent iff there is a word which fulfills one of them but not the other) in $\mathsf{FO}^2[\text{symbols}]$ with quantifier depth at most r. Then denote for every word $a \in \Sigma_m^+$ the set of all formulas in $\mathsf{FO}^2[\text{symbols}]$ with quantifier depth at most r, which are fulfilled by a, with $[a]_r^2$. If for two words a and b we have $a \ncong_r^2 b$, i.e. $[a]_{\cong_r^2} \neq [b]_{\cong_r^2}$, then we know by the above quoted theorem that also $[a]_r^2 \neq [b]_r^2$. And since there are only finitely many pairwise distinct sets of the form $[a]_r^2$, there are also only finitely many classes of $\cong_r^2$.

Now let

$$L = \bigcup_{i \in I} [a_i]_{\cong_r^2} \quad \text{with } [a_i]_{\cong_r^2} \neq [a_j]_{\cong_r^2} \text{ for all } i \neq j.$$

The above yields that I is finite and that $[a_i]_r^2$ is finite for all $i \in I$. Therefore we can construct a formula $\varphi \in \mathsf{FO}^2$[symbols] with quantifier depth at most r and $L = \mathcal{L}_\varphi$ as follows:

$$\varphi := \bigvee_{i \in I} \bigwedge_{\psi \in [a_i]_r^2} \psi$$

□

5.2 $\equiv_n$ to Ehrenfeucht-Fraïssé Games

We will now begin a circular argument to prove the equivalence of $(i) - (iii)$ in Theorem 5.3.

Lemma 5.5.
Let $n \in \mathbb{N}$ and $L \subseteq \Sigma_m^+$ a union of classes of $\equiv_n$. Then L is a union of classes of $\cong_{n+m}^2$ (with underlying first order vocabulary $\{<, P_0, P_1, \dots\}$).

Proof. Since by assumption L is a union of classes of $\equiv_n$ it is enough to show that for all $a \in \Sigma_m^+ : [a]_{\cong_{n+m}^2} \subseteq [a]_{\equiv_n}$, i.e. $\cong_{n+m}^2$ is a refinement of $\equiv_n$. And because $[a]_{\cong_{n+m}^2} \subseteq [a]_{\cong_{n+|\lambda(a)|}^2}$ for all $a \in \Sigma_m^+$ (if Duplicator wins the $(n+m)$-round game she even more wins the $(n+|\lambda(a)|)$-round game) it is enough to show that for all $a \in \Sigma_m^+ : \ [a]_{\cong_{n+|\lambda(a)|}^2} \subseteq [a]_{\equiv_n}$. This is equivalent to:

$$\text{For all } a, b \in \Sigma_m^+ : \ a \not\equiv_n b \ \Rightarrow \ a \not\cong_{n+|\lambda(a)|}^2 b.$$

So let $a, b \in \Sigma_m^+$ and $a \not\equiv_n b$. The proof that $a \not\cong_{n+|\lambda(a)|}^2 b$ goes by induction on $n + |\lambda(a)|$. Assume $n = 0$. This is not possible because $a \equiv_0 b$ for all $a, b \in \Sigma_m^+$.

So we get $n > 0$. If $\lambda(a) \neq \lambda(b)$ we immediately have $a \not\cong_1^2 b$ since Spoiler has a trivial winning strategy: He chooses a structure that has a letter not occuring in the other word and pebbles a position with this letter. If $\lambda(a) = \lambda(b) \neq \emptyset$ (the case $|\lambda(a)| = |\lambda(b)| = 0$ cannot occur since it implies $a = b = \varepsilon$ and therefore $a \equiv_n b$) there is by definition of $\equiv_n$ a $p \in \lambda(a)$ such that $a' \not\equiv_n b'$ or $a'' \not\equiv_{n-1} b''$ if $(a', p, a'') = p\text{-ldecomp}(a)$ and $(b', p, b'') = p\text{-ldecomp}(b)$; or the analog thing happens for the p-right decompositions but w.l.o.g. we can assume it happens for the p-left decompositions.

If $a' \not\equiv_n b'$ we get $n + |\lambda(a')| < n + |\lambda(a)|$ since $p \notin \lambda(a')$ and by induction hypothesis this leads to $a' \not\cong_{n+|\lambda(a')|}^2 b'$. This means that Spoiler has a winning strategy for the $(n + |\lambda(a')|)$-round game on a' and b'. For the $(n + |\lambda(a)|)$-round game on a and b he just plays this strategy and either wins or w.l.o.g. Duplicator must answer with setting $\alpha_s(x_1) := k$ for some $s \leq n + |\lambda(a')|, \ k \geq |a'|$. Then Spoiler sets $\alpha_{s+1}(x_2) := |a'|$. This leads to $\alpha_{s+1}(x_2) \leq \alpha_{s+1}(x_1) = k$ and so Duplicator must make sure $\beta_{s+1}(x_2) \leq \beta_{s+1}(x_1)$. But at the same time she must set $\beta_{s+1}(x_2)$ on a position j with $b_j = p$. But as $\beta_s(x_1) < |a'|$ there is no position which fulfills both conditions and so Duplicator loses in round $s + 1 \leq n + |\lambda(a')| + 1 = n + |\lambda(a)|$.

After round s: $\underbrace{a_0a_1\dots a_{|a'|-1}}_{\alpha_s(x_2)}pa_{|a'|+1}\cdots\underbrace{a_k}_{\alpha_s(x_1)}\dots a_{|a|-1}$

$\underbrace{b_0b_1\dots b_{|b'|-1}}_{\beta_s(x_2)<\beta_s(x_1)}pb_{|b'|+1}\dots b_{|b|-1}$

After round $s+1$: $a_0a_1\dots a_{|a'|-1}\ \underbrace{p}_{\alpha_{s+1}(x_2)}\ a_{|a'|+1}\cdots\ \underbrace{a_k}_{\alpha_{s+1}(x_1)}\ \dots a_{|a|-1}$

$\underbrace{b_0b_1\dots b_{|b'|-1}}_{\beta_{s+1}(x_1)}\ \underbrace{pb_{|b'|+1}\dots b_{|b|-1}}_{\beta_{s+1}(x_2)\text{Contradiction!}}$

If $a'' \not\equiv_{n-1} b''$ we get $n-1+|\lambda(a)| < n+\lambda(a)$ and by induction hypothesis this leads to $a'' \not\equiv^2_{n-1+|\lambda(a)|} b''$. Analog to the first case this means that Spoiler has a winning strategy for the $(n-1+|\lambda(a)|)$-round game on a'' and b''. And again for the game on a and b he just plays this strategy and either wins or w.l.o.g. Duplicator must answer with setting $\alpha_s(x_1) := k$ for some $s \le n-1+|\lambda(a)|$, $k \le |a'|$. Then Spoiler sets $\beta_{s+1}(x_2) := |a'|$. This leads to $\beta_{s+1}(x_2) < \beta_{s+1}(x_1)$ and so Duplicator must make sure $\alpha_{s+1}(x_2) < \alpha_{s+1}(x_1)$. But at the same time she must set $\alpha_{s+1}(x_2)$ on a position j with $a_j = p$. But as $\alpha_s(x_1) \le |a'|$ there is no position which fulfills both conditions and so Duplicator loses in round $s+1 \le n-1+|\lambda(a)|+1 = n+|\lambda(a)|$.

After round s: $a_0a_1\dots\underbrace{a_k}_{\alpha_s(x_1)}\dots a_{|a'|-1}p\underbrace{a_{|a'|+1}\dots a_{|a|-1}}_{\alpha_s(x_2)}$

$b_0b_1\dots b_{|b'|-1}p\underbrace{b_{|b'|+1}\dots b_{|b|-1}}_{\beta_s(x_1)<\beta_s(x_2)}$

After round $s+1$: $a_0a_1\dots\underbrace{a_k}_{\alpha_{s+1}(x_1)}\dots a_{|a'|-1}\ \underbrace{pa_{|a'|+1}\dots a_{|a|-1}}_{\alpha_{s+1}(x_2)\text{Contradiction!}}$

$b_0b_1\dots b_{|b'|-1}\ \underbrace{p}_{\beta_{s+1}(x_2)}\ \underbrace{b_{|b'|+1}\dots b_{|b|-1}}_{\beta_{s+1}(x_1)}$

□

5.3 Ehrenfeucht-Fraïssé Games to Finite Semigroups

To prove the next part of our circular argument we need some additional semigroup theory. However, what we need is very basic and is only needed for technical reasons at the beginning of our main proof.

Definition 5.6. (Divisor)
Let S, S' be arbitrary semigroups. Then we say that S *divides* S' iff S is a homomorphic image of a subsemigroup of S'. We write $S \prec S'$.

Lemma 5.7.
Let $\sim$ be an arbitrary congruence relation on Σ_m^+ and let $L \subseteq \Sigma_m^+$ be a finite union of

classes of $\sim$. Then it follows that $\Sigma_m^+/_\sim$ is divided by $\Sigma_m^+/_{\approx_L}$, the syntactic semigroup of L.

Proof. This is a technical but fairly easy proof found e.g. in [Str94, p. 56] by Straubing. □

We will now give the main proof.

Lemma 5.8.
Let $r \in \mathbb{N}$ and $L \subseteq \Sigma_m^+$ a union of classes of $\cong_r^2$. Then the syntactic semigroup of L is in **DA**.

Proof. Because L is a finite union of elements of $\Sigma_m^+/_{\cong_r^2}$ (the union is finite because there are only finitely many equivalence classes of $\cong_r^2$ since every language recognized by an FO^2-sentence with quantifier depth at most r it follows by Lemma 5.7 that $\Sigma_m^+/_{\cong_r^2}$ is divided by $\Sigma_m^+/_{\approx_L}$. Now let φ be the corresponding homomorphism from a subsemigroup of $\Sigma_m^+/_{\cong_r^2}$ onto $\Sigma_m^+/_{\approx_L}$. We have to show that for all $x, y, z \in \Sigma_m^+/_{\approx_L}$ the condition $(xyz)^\omega y(xyz)^\omega = (xyz)^\omega$ holds true.

First note that if $S \prec S'$ are semigroups and ψ the corresponding homomorphism from a subsemigroup of S' onto S we have $\psi(x^\omega) = (\psi(x))^\omega$ for all $x \in S'$. This follows immediately from the uniqueness part of Remark 2.4 since $\psi(x^\omega) = \psi(x^n)$ for an $n \in \mathbb{N}$ and $(\psi(x))^n = \psi(x^n) = \psi(x^{2n}) = (\psi(x))^{2n}$.

Now let $x', y', z' \in \Sigma_m^+/_{\approx_L}$. Then there are $x, y, z \in \Sigma_m^+/_{\cong_r^2}$ with

$$\varphi(x) = x', \varphi(y) = y' \text{ and } \varphi(z) = z'.$$

If $(xyz)^\omega y(xyz)^\omega = (xyz)^\omega$ we get

$$(x'y'z')^\omega y'(x'y'z')^\omega = \varphi((xyz)^\omega y(xyz)^\omega) = \varphi((xyz)^\omega) = (x'y'z')^\omega.$$

Therefore it is enough to show that

$$(xyz)^\omega y(xyz)^\omega = (xyz)^\omega.$$

Since there is an $n > r$ with $(xyz)^\omega = (xyz)^n = (xyz)^{2n}$ it is enough to show that $(xyz)^n y(xyz)^n = (xyz)^{2n}$ for all $n > r$. Therefore we have to show that Duplicator wins the r-round two-pebble Ehrenfeucht-Fraïssé game on $u = (abc)^n b(abc)^n$ and $v = (abc)^{2n}$ for arbitrary $a, b, c \in \Sigma_m^+$ with $[a]_{\cong_r^2} = x, [b]_{\cong_r^2} = y$ and $[c]_{\cong_r^2} = z$.

We may assume that $a, b, c \in \Sigma_m$ and that they are pairwise distinct from each other. The single letter assumption stems from the fact that if one of the words was longer, Duplicator could simply play the strategy she has for the one letter case, i.e. always pebble the same occurrence of a, b or c as she would in the one letter case, and to decide which letter inside the respectively chosen block she has to pebble she would simply always pebble the same inner-block position as Spoiler just did. The pairwise distinctness can be assumed because in the case of an equality the words u and v would be more alike than in the pairwise distinct case. Therefore Duplicator could still play her strategy for the latter case to win the game even more.

The first and the last $3n$ positions in u correspond naturally to the ones in v. Duplicator's winning strategy is to place the pebble on the corresponding position in the other word if possible and if not to place it as close as possible to the corresponding position.

In the following we will write k' for the position in v which corresponds to the position k in u, i.e. $k' := \begin{cases} k & \text{iff } k < 3n \\ k-1 & \text{iff } k > 3n \end{cases}$. Note that k' does not exist if $k = 3n$.

We now prove the following claim from which it is apparent that Duplicator wins the r-round game.

Claim. Duplicator has a strategy which ensures that

$$\begin{array}{lll} \text{for all } s \leq r: & \text{For all } i \in \{1,2\} \text{ (with } \alpha_s(x_i) \text{ and } \beta_s(x_i) \text{ defined)}: & \\ & \alpha_s(x_i)' = \beta_s(x_i) & (i) \\ & \text{or} \quad \alpha_s(x_i)', \beta_s(x_i) \in \{3(n-s), \ldots, 3(n+s)\} & (ii) \\ & \text{and} \quad \beta_s \circ \alpha_s^{-1} \text{ is an isomorphism.} & \end{array}$$

Proof of the claim by induction on s. Let $s = 0$. Then the claim is trivially true since $\alpha_0(x_i)$ and $\beta_0(x_i)$ are undefined for $i = 1, 2$.

Now let the claim be true for $s \geq 0$. If only one of the two pebbles will be in the game after round $s+1$, i.e. only one of $\alpha_{s+1}(x_1)$ and $\alpha_{s+1}(x_2)$ is defined, the strategy for Duplicator to ensure the claim for round $s+1$ is quite trivial. So assume that both will be defined and therefore $s \geq 1$. Also assume w.l.o.g. that in round $s+1$ Spoiler sets pebble x_1. We have to consider two cases now.

The first one is that for $i = 2$ condition (ii) is not satisfied for round s. In this case condition (i) must be satisfied for round s instead; and then automatically also for round $s+1$ because pebble x_2 is not moved in round $s+1$. If Spoiler now sets $\alpha_{s+1}(x_1) := k$ with $k \neq 3n$ Duplicator can simply answer with setting $\beta_{s+1}(x_1) := k'$ and if he sets $\beta_{s+1}(x_1) := k'$ she can set $\alpha_{s+1}(x_1) := k$. In this subcase condition (i) is satisfied for $i = 1$ and round $s+1$.

$$\begin{array}{lccccccc} u: & abc & \overset{\alpha_s(x_2)}{\overset{\uparrow}{\ldots}} & abc & b & abc & \overset{\alpha_{s+1}(x_1):=k}{\overset{\uparrow}{\ldots}} & abc \\ v: & abc & \overset{\beta_s(x_2)=\alpha_s(x_2)'}{\overset{\uparrow}{\ldots}} & abc & & abc & \overset{\beta_{s+1}(x_1):=k'}{\overset{\uparrow}{\ldots}} & abc \end{array}$$

If Spoiler sets $\alpha_{s+1}(x_1) := 3n$, i.e. he pebbles the b in the center position, Duplicator can answer with setting $\beta_{s+1}(x_1) := 3n-2$. In this subcase condition (ii) is satisfied for $i = 1$ and round $s+1$ because $3n-2 \in \{3(n-(s+1)), \ldots, 3(n+(s+1))\}$ for all $s \geq 0$.

$$\begin{array}{lccccccc} u: & abc & \overset{\alpha_s(x_2)}{\overset{\uparrow}{\ldots}} & a & b & c & \overset{\alpha_{s+1}(x_1):=3n}{\overset{\uparrow}{b}} & abc \ldots abc \\ v: & abc & \overset{\beta_s(x_2)=\alpha_s(x_2)'}{\overset{\uparrow}{\ldots}} & a & \overset{\beta_{s+1}(x_1):=3n-2}{\overset{\uparrow}{b}} & c & & abc \ldots abc \end{array}$$

The second case is that $\alpha_s(x_2)', \beta_s(x_2) \in \{3(n-s), \ldots, 3(n+s)\}$; and again the same condition is automatically satisfied for $i = 2$ and round $s+1$. If Spoiler sets $\alpha_{s+1}(x_1) := k$

(or $\beta_{s+1}(x_1) := k'$) with $k' \notin \{3(n-s), \ldots, 3(n+s)\}$ Duplicator can easily answer by setting $\beta_{s+1}(x_1) := k'$ (or respectively $\alpha_{s+1}(x_1) := k$) and thereby fulfill condition (i) for $i = 1$ and round $s+1$.

$$\begin{array}{lccccccccccccc}
u: & abc & \ldots & abc & \underset{\uparrow}{\ldots}_{\alpha_s(x_2)} & abc & b & abc & \ldots & abc & \underset{\uparrow}{\ldots}_{\alpha_{s+1}(x_1):=k} & abc \\
v: & abc & \ldots & abc & \ldots & abc & & abc & \underset{\uparrow}{\ldots}_{\beta_s(x_2)} & abc & \underset{\uparrow}{\ldots}_{\beta_{s+1}(x_1):=k'} & abc
\end{array}$$

$$\overbrace{\qquad\qquad\qquad\qquad\qquad\qquad}^{\{3(n-s),\ldots,3(n+s)\}}$$

If $k' \in \{3(n-s), \ldots, 3(n+s)\}$ it may not be possible for Duplicator to simply pebble the corresponding position; but this is not necessary to fulfill condition (ii). So if Spoiler sets $\alpha_{s+1}(x_1) := k$ with $k' \in \{3(n-s), \ldots, 3(n+s)\}$ and $k > \alpha_s(x_2)$ she will respond by setting $\beta_{s+1}(x_1)$ to the smallest position j with $j > \beta_s(x_2)$ and $v_j = u_k$. If $k < \alpha_s(x_2)$ she will choose the largest position j with $j < \beta_s(x_2)$ and $v_j = u_k$ and if $k = \alpha_s(x_2)$ she sets $\beta_{s+1}(x_1) := \beta_s(x_2)$.

If Spoiler sets $\beta_{s+1}(x_1) := k'$ she acts accordingly. In all subcases she ensures $\alpha_{s+1}(x_1)', \beta_{s+1}(x_1) \in \{3(n-(s+1)), \ldots, 3(n+(s+1))\}$ because by induction hypothesis we have $\alpha_s(x_2)', \beta_s(x_2) \in \{3(n-s), \ldots, 3(n+s)\}$ and the position pebbled by Duplicator with x_1 is the closest position to x_2 with the correct letter on it and can therefore be at most three letters away from the position of x_2 (except when x_2 is very close to the center where it can be a distance of four; but clearly condition (ii) is still fulfilled then).

$$\begin{array}{lccccccccccccc}
u: & abc & \ldots & abc & \underset{\uparrow}{\ldots}_{\alpha_s(x_2)} & \underset{\uparrow}{a}_{\alpha_{s+1}(x_1):=k} & bc & b & abc & \ldots & a & bc & \ldots & abc \\
v: & abc & \ldots & abc & \ldots & a & bc & & abc & \underset{\uparrow}{\ldots}_{\beta_s(x_2)} & \underset{\uparrow}{a}_{\beta_{s+1}(x_1):=k'} & bc & \ldots & abc
\end{array}$$

$$\overbrace{\qquad\qquad\qquad\qquad}^{\{3(n-s),\ldots,3(n+s)\}}$$

$$\overbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}^{\{3(n-(s+1)),\ldots,3(n+(s+1))\}}$$

□

5.4 Finite Semigroups to $\equiv_n$

For the next proof we need some more semigroup related notation. Although it may not seem as basic as in the last section it is in fact still very basic and just not as commonly used.

Definition 5.9.
For a semigroup S and $a, b \in S$ we write $a \leq_{\mathcal{R}} b$ iff $\{a\} \cup \{as \mid s \in S\} \subseteq \{b\} \cup \{bs \mid s \in S\}$ and $a \mathcal{R} b$ iff $a \leq_{\mathcal{R}} b$ and $b \leq_{\mathcal{R}} a$. $\leq_{\mathcal{L}}$ is defined analogously by $a \leq_{\mathcal{L}} b$ iff $\{a\} \cup \{sa \mid s \in S\} \subseteq \{b\} \cup \{sb \mid s \in S\}$.

Remark 5.10.
If $L \subseteq \Sigma_m^+$ and $a, b \in \Sigma_m^+$ then $[ab]_{\approx_L} \leq_{\mathcal{R}} [a]_{\approx_L}$.

Proof. This follows immediately because $\approx_L$ is a congruence relation with respect to the operation of concatenation of words. □

For the sake of clearness we will often write $[a]$ instead of $[a]_{\approx_L}$ in the following.

Definition 5.11.
Let $L \subseteq \Sigma_m^+$ and $u \in \Sigma_m^+$. Then we write $\mathcal{R}\text{decomp}_L(u)$ for the $\mathcal{R}$ *decomposition* of u in L which is defined as the unique sequence $(v_0, p_1, v_1, \ldots, p_l, v_l)$ satisfying the following conditions:

- $u = v_0 p_1 v_1 \ldots p_l v_l$.
- For all $i \in \{1, \ldots, l\}$: $p_i \in \Sigma_m$ and for all $i \in \{0, \ldots, l\}$: $v_i \in \Sigma_m^*$.
- $[u_0]\mathcal{R}[v_0]$ (where u_0 is the first letter of u) and for all $i \in \{1, \ldots, l\}$: $[v_0 p_1 v_1 \ldots p_i]\mathcal{R}[v_0 p_1 v_1 \ldots p_i v_i]$.
- For all $i \in \{1, \ldots, l\}$: $[v_0 p_1 v_1 \ldots v_{i-1}] \not\mathcal{R} [v_0 p_1 v_1 \ldots v_{i-1} p_i]$.

So to construct $\mathcal{R}\text{decomp}_L(u)$ one has to consider the $\approx_L$ classes of increasingly long prefixes of u and whenever a class is not $\mathcal{R}$-equivalent to the class before we get a new p_i, i.e. the p_i are the "breakpoints" where a new $\mathcal{R}$-equivalence class is entered and stayed into until p_{i+1}.

Note that in the definition we could have written $\leq_{\mathcal{R}}$ wherever $\mathcal{R}$ occurs, as the opposite direction, $\geq_{\mathcal{R}}$, automatically holds by Remark 5.10.

The next Lemma contains the most important fact about $\mathcal{R}$ decompositions. We will use it in the proof of Proposition 5.13 to connect $\mathcal{R}$ decompositions and iterations of p-left decompositions.

Lemma 5.12.
Let $L \subseteq \Sigma_m^+$, $\Sigma_{m/\approx_L}^+ \in \mathbf{DA}$, $u \in \Sigma_m^+$ and $\mathcal{R}\text{decomp}_L(u) = (v_0, p_1, v_1, \ldots, p_l, v_l)$. Then $p_{i+1} \notin \lambda(v_i)$ for all $0 \leq i \leq l-1$.

Proof. Let $i \in \{0, \ldots, l-1\}$, $v_i = v_{i,0} v_{i,1} \ldots v_{i,|v_i|-1}$, $j \in \{0, \ldots, |v_i| - 1\}$. Then we have to show that $p_{i+1} \neq v_{i,j}$.

Since $[v_0 p_1 v_1 \ldots p_i]\mathcal{R}[v_0 p_1 v_1 \ldots p_i v_i]$ there is $b \in \Sigma_m^+$ with $[v_0 p_1 v_1 \ldots p_i v_i][b] = [v_0 p_1 v_1 \ldots p_i]$. Now we set $w := [v_0 p_1 \ldots p_i v_i]$, $x := [b v_{i,0} v_{i,1} \ldots v_{i,j-1}]$, $y := [v_{i,j}]$ and $z := [v_{i,j+1} v_{i,j+2} \ldots v_{i,|v_i|-1}]$.

We will now show that $[v_0 p_1 v_1 \ldots p_i v_i]\mathcal{R}[v_0 p_1 v_1 \ldots p_i v_i v_{i,j}]$, i.e. $w\mathcal{R}wy$, which immediately yields $p_{i+1} \neq v_{i,j}$ because $[v_0 p_1 v_1 \ldots v_i] \not\mathcal{R} [v_0 p_1 v_1 \ldots v_i p_{i+1}]$. Because $\Sigma_{m/\approx_L}^+ \in \mathbf{DA}$ there is an $n \in \mathbb{N} \setminus \{0\}$ with $(xyz)^n y (xyz)^n = (xyz)^n$. This yields $w(xyz)^n y (xyz)^n = w(xyz)^n$. On the other hand we have

$$\begin{aligned} w(xyz)^n &= [v_0 p_1 v_1 \ldots p_i v_i]([b v_{i,0} v_{i,1} \ldots v_{i,j-1}][v_{i,j}][v_{i,j+1} \ldots v_{i,|v_i|-1})])^n \\ &= [v_0 p_1 v_1 \ldots p_i v_i \, (b v_i)^n] \\ &= [v_0 p_1 v_1 \ldots p_i v_i \, b (v_i b)^{n-1} v_i] \\ &\overset{\text{Def. von } b}{=} [v_0 p_1 v_1 \ldots p_i \; v_i] \end{aligned}$$

$$\begin{aligned}\text{and } w(xyz)^n y(xyz)^n &= w(xyz)^n[v_{i,j}]([bv_{i,0}v_{i,1}\dots v_{i,j-1}][v_{i,j}][v_{i,j+1}\dots v_{i,|v_i|-1})])^n\\ &\overset{\text{see above}}{=} [v_0p_1v_1\dots p_iv_i][v_{i,j}][(bv_i)^n]\\ &= [v_0p_1v_1\dots p_iv_iv_{i,j}][(bv_i)^n].\end{aligned}$$

This leads to $[v_0p_1v_1\dots p_iv_iv_{i,j}][(bv_i)^n] = [v_0p_1v_1\dots p_iv_i]$ and therefore to $[v_0p_1v_1\dots p_iv_iv_{i,j}] \geq_{\mathcal{R}} [v_0p_1v_1\dots p_iv_i]$. Since by Remark 5.10 $[v_0p_1v_1\dots p_iv_iv_{i,j}] \leq_{\mathcal{R}} [v_0p_1v_1\dots p_iv_i]$ we have $[v_0p_1v_1\dots p_iv_iv_{i,j}]\mathcal{R}[v_0p_1v_1\dots p_iv_i]$. □

The following is the main result of this section. The proof may not be the longest but is probably the most involved and difficult one of this whole chapter.

Proposition 5.13.
Let $L \subseteq \Sigma_m^+$ and the syntactic semigroup of L in **DA**. *Then there is an $n \in \mathbb{N}$ such that L is a union of classes of $\equiv_n$.*

Proof. Since L is a union of classes of $\approx_L$ it is enough to show that there is an n such that for all $a, b \in \Sigma_m^+ :\ a \equiv_n b \ \Rightarrow\ a \approx_L b$.

In [Pin86, p. 65] by Pin it is proved that if $S \in \mathbf{A} \supset \mathbf{DA}$, $a, b \in S$, $a\mathcal{R}b$ and $a\mathcal{L}b$ then $a = b$. We will now show by induction on $|\lambda(a)|$ that from $a \equiv_k b$ for $k > |\lambda(a)||\Sigma^+_{m/\approx_L}|$ it follows that $[a]_{\approx_L} = [b]_{\approx_L}$. Therefor we will only show that $[a]_{\approx_L} \geq_{\mathcal{R}} [b]_{\approx_L}$ and then get by a-b-symmetry that $[a]_{\approx_L} \leq_{\mathcal{R}} [b]_{\approx_L}$ and therefore also $[a]_{\approx_L}\mathcal{R}[b]_{\approx_L}$. By $\mathcal{R}$-$\mathcal{L}$-symmetry we immediately get $[a]_{\approx_L}\mathcal{L}[b]_{\approx_L}$. And from the above that $a \approx_L b$. Choosing $n := m|\Sigma^+_{m/\approx_L}| + 1$ ensures that $n > |\lambda(a)||\Sigma^+_{m/\approx_L}|$ for every $a \in \Sigma_m^+$ and so yields the wanted result.

If $|\lambda(a)| = 0$ we have nothing to show since $a \equiv_1 b$ and therefore $\lambda(a) = \lambda(b)$ and $a = b = \varepsilon$. If $|\lambda(a)| > 0$ let $(a_0, p_1, a_1, \dots, p_l, a_l) := \mathcal{R}\text{decomp}_L(a)$. Lemma 5.12 yields $p_{i+1} \notin a_i$ and from this follows (simply by definition of p-left decompositions) that $(a_i, p_{i+1}, a_{i+1}\dots p_la_l) = p_{i+1}\text{-ldecomp}(a_ip_{i+1}\dots a_l)$ for every $i \in \{0, \dots, l-1\}$. We now inductively define b_i, b'_i for every $i \in \{0, \dots, l\}$. First let $b'_0 := b$. Now for every $i \in \{0, \dots, l-1\}$ let $b_i, b'_{i+1} \in \Sigma_m^+$ with $p_{i+1}\text{-ldecomp}(b'_i) = (b_i, p_{i+1}, b'_{i+1})$. And finally let $b_l := b'_l$.

Since $a \equiv_k b$ we get that $a_0 \equiv_k b_0$ and $a_1p_2a_2\dots p_la_l \equiv_{k-1} b'_1 = b_1p_2b_2\dots p_lb_l$. Recursively follows that $a_i \equiv_{k-i} b_i$ for every $i \in \{0, \dots, l\}$ ($l < k$ is proven below). Since $p_{i+1} \notin a_i$ it follows that $|\lambda(a_i)| < |\lambda(a)|$ for every $i \in \{0, \dots, l-1\}$ (not for $i = l$ since there is no p_{l+1}). From this also follows

$$k - i > k - l > |\lambda(a)||\Sigma^+_{m/\approx_L}| - |\Sigma^+_{m/\approx_L}| = (|\lambda(a)| - 1)|\Sigma^+_{m/\approx_L}| \geq |\lambda(a_i)||\Sigma^+_{m/\approx_L}|$$

for every $i \in \{0, \dots, l-1\}$ since $l < |\Sigma^+_{m/\approx_L}|$ because l is less than the number of $\mathcal{R}$ classes of $\Sigma^+_{m/\approx_L}$ and the number of $\mathcal{R}$ classes of a semigroup can obviously not be higher than the number of elements of that semigroup.

So we can use the induction hypothesis and get that $[a_i] = [b_i]$ for every $i \in \{0, \dots, l-1\}$. Therefore we have

$$[a] \overset{\text{Def. of } \mathcal{R}\text{decomp}_L(a)}{\mathcal{R}} [a_0p_1a_1\dots p_l] \overset{[a_i]=[b_i]}{=} [b_0p_1b_1\dots p_l] \overset{\text{Remark 5.10}}{\geq_{\mathcal{R}}} [b_0p_1\dots p_lb_l] = [b].$$

□

6 First Order Logic and Local Testability

In this chapter we will introduce a combinatorial characterization for formal language classes. This characterization is called local threshold testability and is a generalization of local testability which goes back to [MP71] and has received a lot of attention since.

Definition 6.1.
Let $a \in \Sigma_m^+$, $b \in \Sigma_m^*$. Then $\mathrm{Fact}(a,b)$ denotes the number of times a occurs in b, i.e. $\mathrm{Fact}(a,b) := |\{i \mid i \in \{0,\ldots,|b|-1\} \text{ and } b_i b_{i+1} \ldots b_{i+|a|-1} = a\}|$. For example, $\mathrm{Fact}(pp, qpqppqppp) = 3$.

We now define an equivalence relation $\sim_k^r$ on Σ_m^+.

Definition 6.2.
Let $a, b \in \Sigma_m^*$. Then $a \sim_k^r b$ iff

$$
\begin{array}{llll}
 & |a| \le k \text{ and } a = b & & \\
\text{or} & |a| > k \text{ and } & a_0 \ldots a_{k-1} = b_0 \ldots b_{k-1}, & \\
 & & \quad \text{i.e. the prefixes of } a \text{ and } b \text{ of length } k \text{ are equal.} & \\
 & & \text{and} \quad a_{|a|-k} \ldots a_{|a|-1} = b_{|b|-k} \ldots b_{|b|-1}, & \\
 & & \quad \text{i.e. the suffixes of } a \text{ and } b \text{ of length } k \text{ are equal.} & \\
 & & \text{and} \quad \text{for all } c \in \Sigma_m^+ \text{ with } |c| \le k : & \\
 & & \quad \mathrm{Fact}(c,a) = \mathrm{Fact}(c,b) < r & \\
 & & \text{or} \quad \mathrm{Fact}(c,a) \ge r \text{ and } \mathrm{Fact}(c,b) \ge r. &
\end{array}
$$

Note that for arbitrary but fixed k, r there are obviously only finitely many pairwise distinct equivalence classes of $\sim_k^r$.

Definition 6.3.
Let $L \subseteq \Sigma_m^+$. Then L is called *locally threshold testable with threshold r* iff L is a union of equivalence classes of $\sim_k^r$ for a $k \in \mathbb{N}$.
L is called *locally threshold testable* iff it is locally threshold testable with threshold r for an $r \in \mathbb{N}$.
And L is called *locally testable* iff it is locally threshold testable with threshold 1.

The name "locally testable" comes from the fact that for a locally testable language one can check whether a given word belongs to the language by simply scanning the word with a fixed-size window and then only considering which words occured in the window (without respect to the order or cardinality of these words). The addition "threshold" stems from the fact that one no longer only considers the set of all occurring strings but also their cardinalities up to the specified threshold.

We will now quote an equivalence result between local threshold testability and a certain first order logic characterization and then prove Theorem 3.4 $(i) \Leftrightarrow (iii)$ which is a new equivalence result for a subclass of all locally threshold testable languages.

Theorem 6.4.
Let $L \subseteq \Sigma_m^+$. Then $L = \mathcal{L}_\varphi$ for a $\varphi \in \mathsf{FO}[\mathrm{succ}]$ iff L is locally threshold testable.

Proof. This is Theorem IV.3.3 in [Str94, p. 49]. □

Theorem 6.5.
Let $L \subseteq \Sigma_m^+$. Then $L = \mathcal{L}_\varphi$ for a $\varphi \in \mathsf{FO}^2[\mathrm{succ}]$ iff L is locally threshold testable with threshold 2.

Proof. We modify and refine the proof from [Str94, p. 49].

"$\Leftarrow$": The proof of this direction is much like the proof from [Str94] – only slightly modified to handle the problem with the restricted number of variables which is quite easy to cope with in this part.

Let $k \in \mathbb{N}$ such that L is the union of $\sim_k^2$ classes. It is enough to show that for all $a \in \Sigma_m^+$ there is a $\psi_a \in \mathsf{FO}^2[\mathrm{succ}]$ with $\mathcal{L}_{\psi_a} = [a]_{\sim_k^2}$ because then follows

$$L = \mathcal{L}_\varphi \text{ for } \varphi := \bigvee_{[a]_{\sim_k^2} \subseteq L} \psi_a.$$

So let $a \in \Sigma_m^+$ and $a = a_0 a_1 \dots a_l$. In the following we will use capital letters with indices (A_0, C_0 etc.) to denote the unary predicate symbols corresponding to the letters from Σ_m represented by the lowercase letters with the same indices (a_0, c_0 etc.). If $|a| \leq k$ we can simply set

$$\psi_a := \exists x \Big(\neg\exists y(\mathrm{succ}(y,x)) \wedge A_0(x) \wedge \underbrace{\exists y \Big(\mathrm{succ}(x,y) \wedge A_1(y) \wedge \exists x \big(\mathrm{succ}(y,x) \wedge A_2(x) \wedge \exists y \dots \big) \Big)}_{l \text{ quantifiers}} \Big).$$

If $|a| > k$ the situation is a little bit more complex and we need for arbitrary $c = c_0 c_1 \dots c_j \in \Sigma_m^+$ the formulas

$$\text{“Fact}(c,\cdot) \geq 1\text{”} := \underbrace{\exists x \Big(C_0(x) \wedge \exists y \big(\mathrm{succ}(x,y) \wedge C_1(y) \wedge \exists x \big(\mathrm{succ}(y,x) \wedge C_2(x) \wedge \exists y \dots \big) \big) \Big)}_{j+1 \text{ quantifiers}}$$

and

$$\begin{aligned} \text{“Fact}(c,\cdot) \geq 2\text{”} := \quad \exists x \Big(C_0(x) \wedge \underbrace{\exists y \big(\mathrm{succ}(x,y) \wedge C_1(y) \wedge \exists x \big(\mathrm{succ}(y,x) \wedge C_2(x) \wedge \exists y \dots \big) \big)}_{j \text{ quantifiers}} \\ \underbrace{\wedge\ \exists y \big(\neg\, x = y \wedge C_0(y) \wedge \exists x \big(\mathrm{succ}(y,x) \wedge C_1(x) \wedge \exists y \dots \big) \big)}_{j+1 \text{ quantifiers}} \Big). \end{aligned}$$

Then follows that ψ_a is the conjunction of the following formulas:

- $\exists x\Big(\neg\exists y(\mathrm{succ}(y,x))\wedge A_0(x)\wedge\underbrace{\exists y\Big(\mathrm{succ}(x,y)\wedge A_1(y)\wedge\exists x\big(\mathrm{succ}(y,x)\wedge A_2(x)\wedge\exists y\ldots\big)\Big)}_{k-1\text{ quantifiers}}\Big)$,

  which ensures that a word $b\in\Sigma_m^+$ with $b\models\psi_a$ has the same prefix of length k as a.

- $\exists x\Big(\neg\exists y(\mathrm{succ}(x,y))\wedge A_l(x)\wedge\underbrace{\exists y\Big(\mathrm{succ}(y,x)\wedge A_{l-1}(y)\wedge\exists x\big(\mathrm{succ}(x,y)\wedge A_{l-2}(x)\wedge\exists y\ldots\big)\Big)}_{k-1\text{ quantifiers}}\Big)$,

  which ensures that a word $b\in\Sigma_m^+$ with $b\models\psi_a$ has the same suffix of length k as a.

- for every $c\in\Sigma_m^+$ with $|c|\leq k$ and $\mathrm{Fact}(c,a)=0$ the formula $\neg$"$\mathrm{Fact}(c,\cdot)\geq 1$".

- for every $c\in\Sigma_m^+$ with $|c|\leq k$ and $\mathrm{Fact}(c,a)=1$ the formula "$\mathrm{Fact}(c,\cdot)\geq 1$" $\wedge\neg$"$\mathrm{Fact}(c,\cdot)\geq 2$".

- for every $c\in\Sigma_m^+$ with $|c|\leq k$ and $\mathrm{Fact}(c,a)\geq 2$ the formula "$\mathrm{Fact}(c,\cdot)\geq 2$".

"$\Rightarrow$": We show that for all $r\in\mathbb{N}, a,b\in\Sigma_m^+$: $a\sim^2_{8r+1}b\ \Rightarrow\ a\cong^2_r b$ (with first order vocabulary $\{\mathrm{succ},P_0,P_1,\ldots\}$) which proves that $\sim^2_{8r+1}$ is a refinement of $\cong^2_r$ for every $r\in\mathbb{N}$. The hypothesis now follows immediately: If $L=\mathcal{L}_\varphi$ for a $\varphi\in\mathsf{FO}^2[\mathrm{succ}]$ then by Lemma 5.4 L is the union of classes of $\cong^2_r$ for an $r\in\mathbb{N}$ and therefore L is locally threshold testable with threshold 2 because it is a union of classes of $\sim^2_{8r+1}$ by the above.

Now let $r\in\mathbb{N}, a,b\in\Sigma_m^+$ and $a\sim^2_{8r+1}b$. We will prove the following claim from which it is immediate that Duplicator has a winning strategy in the r-round two-pebble game on a and b with succ as the only additional predicate.

Claim. There is a strategy for Duplicator which ensures that

$$\begin{aligned}\text{for all } s\leq r:\quad &\beta_s\circ\alpha_s^{-1}\text{ is an isomorphism}\\ &\text{and for all } i\in\{1,2\}\text{ such that }\alpha_s(x_i)\text{ and }\beta_s(x_i)\text{ are defined :}\\ &\quad a[\alpha_s(x_i)-4(r-s),\alpha_s(x_i)+4(r-s)]=b[\beta_s(x_i)-4(r-s),\beta_s(x_i)+4(r-s)],\end{aligned}$$

where $a[i,j]:=a_{\max\{i,0\}}\ldots a_{\min\{j,|a|-1\}}$.

Proof of the claim by induction on s. First let $s=0$. Then we have nothing to show. Now let $s=1$. Then only one kind of pebble was already used in the game, say w.l.o.g. that Spoiler set $\alpha_1(x_1)$. Then we have

$$\mathrm{Fact}(a[\alpha_1(x_1)-4(r-1),\alpha_1(x_1)+4(r-1)],a)\geq 1$$

and therefore also

$$\mathrm{Fact}(a[\alpha_1(x_1)-4(r-1),\alpha_1(x_1)+4(r-1)],b)\geq 1$$

because $a\sim^2_{8r+1}b$. And so Duplicator can set $\beta_1(x_1)$ such that

$$b[\beta_1(x_1)-4(r-1),\beta_1(x_1)+4(r-1)]=a[\alpha_1(x_1)-4(r-1),\alpha_1(x_1)+4(r-1)]$$

and trivially also the first condition, namely that $\beta_1 \circ \alpha_1^{-1}$ is an isomorphism, is satisfied.

$$\text{After round 1:}\quad a_0a_1\ldots \underbrace{\ldots a_{\alpha_1(x_1)}\ldots}_{a[\alpha_1(x_1)-4(r-1),\alpha_1(x_1)+4(r-1)]} \ldots a_{|a|-1}$$
$$b_0b_1\ldots \underbrace{\ldots b_{\beta_1(x_1)}\ldots}_{\substack{b[\beta_1(x_1)-4(r-1),\beta_1(x_1)+4(r-1)]\\ =a[\alpha_1(x_1)-4(r-1),\alpha_1(x_1)+4(r-1)]}} \ldots b_{|b|-1}$$

Now let the claim be true for $s \in \{1,\ldots,r-1\}$. If there is still only one pebble in the game when it is Duplicator's $(s+1)$-th turn she can play the same strategy as in the first round. So assume w.l.o.g. that x_2 is played in round $s+1$, i.e. $\alpha_{s+1}(x_1) = \alpha_s(x_1) = i$ and $\beta_{s+1}(x_1) = \beta_s(x_1) = j$. And that Spoiler set $\alpha_{s+1}(x_2) = k$. Now we have to consider several different cases.

Case 1. If $k \in \{i-1, i, i+1\}$ Duplicator can simply set $\beta_{s+1}(x_2) := j-1$, j or $j+1$ accordingly. Then the isomorphism is preserved because x_2 is pebbled on a neighbour of x_1 and the second condition of the claim is also preserved because by induction hypothesis we have

$$\begin{array}{rl} & a[\alpha_s(x_1)-4(r-s),\alpha_s(x_1)+4(r-s)] \\ = & b[\beta_s(x_1)-4(r-s),\beta_s(x_1)+4(r-s)] \end{array}$$

and since only neighbours of $\alpha_s(x_1)$ and $\beta_s(x_1)$ are pebbled by x_2 in round $s+1$ we get

$$\begin{array}{rl} & a[\alpha_{s+1}(x_2)-4(r-s-1),\alpha_{s+1}(x_2)+4(r-s-1)] \\ = & b[\beta_{s+1}(x_2)-4(r-s-1),\beta_{s+1}(x_2)+4(r-s-1)]. \end{array}$$

$$\text{After round } s\text{:}\quad a_0a_1\ldots \underbrace{\ldots a_{\alpha_s(x_1)}\ldots}_{a[\alpha_s(x_1)-4(r-s),\alpha_s(x_1)+4(r-s)]} \ldots a_{|a|-1}$$
$$b_0b_1\ldots \underbrace{\ldots b_{\beta_s(x_1)}\ldots}_{\substack{b[\beta_s(x_1)-4(r-s),\beta_s(x_1)+4(r-s)]\\ =a[\alpha_s(x_1)-4(r-s),\alpha_s(x_1)+4(r-s)]}} \ldots b_{|b|-1}$$

$$\text{After round } s+1\text{:}\quad a_0a_1\ldots\ldots \underbrace{\ldots \overbrace{\ldots a_{\alpha_s(x_1)}a_{\alpha_{s+1}(x_2)}\ldots}^{a[\alpha_{s+1}(x_2)-4(r-s-1),\alpha_{s+1}(x_2)+4(r-s-1)]} \ldots\ldots}_{a[\alpha_s(x_1)-4(r-s),\alpha_s(x_1)+4(r-s)]} \ldots a_{|a|-1}$$
$$b_0b_1\ldots\ldots \underbrace{\ldots \overbrace{\ldots b_{\beta_s(x_1)}b_{\beta_{s+1}(x_2)}\ldots}^{\substack{b[\beta_{s+1}(x_2)-4(r-s-1),\beta_{s+1}(x_2)+4(r-s-1)]\\ =a[\alpha_{s+1}(x_2)-4(r-s-1),\alpha_{s+1}(x_2)+4(r-s-1)]}} \ldots\ldots}_{\substack{b[\beta_s(x_1)-4(r-s),\beta_s(x_1)+4(r-s)]\\ =a[\alpha_s(x_1)-4(r-s),\alpha_s(x_1)+4(r-s)]}} \ldots b_{|b|-1}$$

Case 2. If $k \notin \{i-1, i, i+1\}$ Duplicator has to make sure that $\beta_{s+1}(x_2) \notin \{j-1, j, j+1\}$ in order to preserve the isomorphism. And she has to find a position l on b with

$$b[l-4(r-s-1), l+4(r-s-1)] = a[k-4(r-s-1), k+4(r-s-1)]$$

in order to preserve the second condition of the claim. Since

$$\mathrm{Fact}(a[k-4(r-s-1)-2, k+4(r-s-1)+2], a) \geq 1$$

it follows

$$\mathrm{Fact}(a[k-4(r-s-1)-2, k+4(r-s-1)+2], b) \geq 1.$$

Let $l' \in \{0, \ldots, |b|-1\}$ such that

$$\begin{array}{rl} & b[l'-4(r-s-1)-2, l'+4(r-s-1)+2] \\ = & a[k-4(r-s-1)-2, k+4(r-s-1)+2]. \end{array}$$

Case 2.1. If there is such an l' with $l' \notin \{j-1, j, j+1\}$ Duplicator can set $\beta_{s+1}(x_2) := l'$ and preserve the claim for round $s+1$.

Case 2.2. Otherwise $l' = j + h$ with $h \in \{-1, 0, 1\}$. Now we are going to show that there must not only be an l' which satisfies the above conditions but also an $l'' \neq l'$ which satisfies the same conditions. And there must be corresponding positions k' and k'' on word a. We will then use this to show that although there might not be a position l' on word b with $l' \notin \{j-1, j, j+1\}$ and the surrounding with radius $4(r-s-1)+2$ of l' equals the surrounding with radius $4(r-s-1)+2$ of k, there must be such a position not in $\{j-1, j, j+1\}$ for which the surroundings of the lesser radius $4(r-s-1)$ are equal. And since this is all Duplicator needs to find to preserve the claim the proof will be finished.

First we notice that

$$\{l'-4(r-s-1)-2, \ldots, l'+4(r-s-1)+2\} \subseteq \{j-4(r-s), \ldots, j+4(r-s)\} \quad \text{(i)}.$$

By induction hypothesis we have

$$b[j-4(r-s), j+4(r-s)] = a[i-4(r-s), i+4(r-s)] \quad \text{(ii)}$$

and this implies

$$\begin{array}{rl} & a[k-4(r-s-1)-2, k+4(r-s-1)+2] \\ = & b[l'-4(r-s-1)-2, l'+4(r-s-1)+2] \\ \overset{(i)}{=} & b[j+h-4(r-s-1)-2, j+h+4(r-s-1)+2] \\ \overset{(ii)}{=} & a[i+h-4(r-s-1)-2, i+h+4(r-s-1)+2]. \end{array}$$

And since $k \notin \{i-1, i, i+1\}$ this leads to

$$\begin{array}{rl} & \mathrm{Fact}(a[k-4(r-s-1)-2, k+4(r-s-1)+2], a) \geq 2. \\ a \sim^2_{\delta_{r+1}} b \Rightarrow & \mathrm{Fact}(a[k-4(r-s-1)-2, k+4(r-s-1)+2], b) \geq 2. \end{array}$$

So let $l'' \in \{0, \ldots, |b|-1\}$ such that $l'' \neq l'$ and

$$\begin{array}{rl} & b[l''-4(r-s-1)-2, l''+4(r-s-1)+2] \\ = & b[l'-4(r-s-1)-2, l'+4(r-s-1)+2]. \end{array}$$

Because we are in case 2.2 we have $l'' \in \{j-1, j, j+1\}$. And therefore it follows as before that

$$a[k-4(r-s-1)-2, k+4(r-s-1)+2] = b[l''-4(r-s-1)-2, l''+4(r-s-1)+2]$$

is completely contained in

$$b[j-4(r-s), j+4(r-s)] = a[i-4(r-s), i+4(r-s)]$$

and hence

$$\mathrm{Fact}(a[k-4(r-s-1)-2, k+4(r-s-1)+2], a) \geq 3$$

since $l'' \in \{j-1, j, j+1\} \setminus \{l'\}$.

At this point our method of going back and forth between factors of a and factors of b and thereby increasing the number of repetitions of the factor

$$a[k-4(r-s-1)-2, k+4(r-s-1)+2]$$

in either word does not work any longer because we cannot follow from

$$\mathrm{Fact}(a[k-4(r-s-1)-2, k+4(r-s-1)+2], a) \geq 3$$

that also

$$\mathrm{Fact}(a[k-4(r-s-1)-2, k+4(r-s-1)+2], b) \geq 3.$$

This is due to the fact that we would need $a \sim^3_{8r+1} b$ instead of only $a \sim^2_{8r+1} b$ to conclude such a thing.

We now have that there are $k', k'' \in \{i-1, i, i+1\}$ with $k' \neq k''$ and

$$\begin{aligned} & a[k-4(r-s-1)-2, k+4(r-s-1)+2] \\ = \; & a[k'-4(r-s-1)-2, k'+4(r-s-1)+2] \\ = \; & a[k''-4(r-s-1)-2, k''+4(r-s-1)+2]. \end{aligned}$$

And we know that there is no position $l \notin \{j-1, j, j+1\}$ with

$$b[l-4(r-s-1)-2, l+4(r-s-1)+2] = a[k-4(r-s-1)-2, k+4(r-s-1)+2]$$

but two positions $l', l'' \in \{j-1, j, j+1\}$ with $l' \neq l''$ and

$$\begin{aligned} & b[l'-4(r-s-1)-2, l'+4(r-s-1)+2] \\ = \; & b[l''-4(r-s-1)-2, l''+4(r-s-1)+2] \\ = \; & a[k-4(r-s-1)-2, k+4(r-s-1)+2]. \end{aligned}$$

Case 2.2.1. If $|l'-l''| = 1$ we can w.l.o.g. assume that $l' = j$ and $l'' = j+1$. Since

$$\begin{aligned} & b[l'-4(r-s-1)-2, l'+4(r-s-1)+2] \\ = \; & b[l''-4(r-s-1)-2, l''+4(r-s-1)+2] \end{aligned}$$

it follows that

$$\begin{array}{rcl} b_{l'-4(r-s-1)-2} & = & b_{l''-4(r-s-1)-2} \\ & \overset{l''=l'+1}{=} & b_{l'-4(r-s-1)-1} \\ & = & \dots \\ & \overset{l''=l'+1}{=} & b_{l'} \\ & = & b_{l''} \\ & \overset{l''=l'+1}{=} & b_{l'+1} \\ & \overset{l''=l'+1}{=} & b_{l'+2} \\ & = & \dots \\ & \overset{l''=l'+1}{=} & b_{l'+4(r-s-1)+2} \\ & = & b_{l''+4(r-s-1)+2} \\ & =: & p \end{array}$$

This implies

$$b[\underbrace{l'-4(r-s-1)-2}_{=j-4(r-s-1)-2},\underbrace{l''+4(r-s-1)+2}_{=j+1+4(r-s-1)+2}] = \underbrace{p^{8(r-s-1)+6}}_{:=p\dots p}$$

and hence

$$\begin{array}{rl} & b[j+2-4(r-s-1), j+2+4(r-s-1)] \\ = & p^{8(r-s-1)+1} \\ = & b[l'-4(r-s-1), l'+4(r-s-1)] \\ = & a[k-4(r-s-1), k+4(r-s-1)]. \end{array}$$

And so Duplicator can set $\beta_{s+1}(x_2) = j+2$.

$$a: \quad \dots \overbrace{p\dots p}^{4(r-s-1)+2} \underset{\substack{\uparrow \\ i=\alpha_{s+1}(x_1)}}{p} \; p \overbrace{p\dots p}^{4(r-s-1)+2} \; \dots\dots \; \overbrace{p\dots p}^{4(r-s-1)+2} \underset{\substack{\uparrow \\ k=\alpha_{s+1}(x_2)}}{p} \overbrace{p\dots p}^{4(r-s-1)+2} \; \dots$$

$$b: \quad \dots \overbrace{p\dots p}^{4(r-s-1)+2} \underset{\substack{\uparrow \\ l'=j=\beta_{s+1}(x_1)}}{p} \underset{\substack{\uparrow \\ l''}}{p} \underset{\substack{\uparrow \\ \beta_{s+1}(x_2)}}{p} \overbrace{p\dots p}^{4(r-s-1)+1} \; \dots$$

Case 2.2.2. If $|l'-l''| = 2$ we can w.l.o.g. assume that $l' = j-1$ and $l'' = j+1$. Since

$$\begin{array}{rl} & b[l'-4(r-s-1)-2, l'+4(r-s-1)+2] \\ = & b[l''-4(r-s-1)-2, l''+4(r-s-1)+2] \end{array}$$

it follows that

$$\begin{aligned} b_{l'-4(r-s-1)-2} &= b_{l''-4(r-s-1)-2} \\ &\overset{l''=l'+2}{=} b_{l'-4(r-s-1)} \\ &= \dots \\ &\overset{l''=l'+2}{=} b_{l'} \\ &= b_{l''} \\ &\overset{l''=l'+2}{=} b_{l'+2} \\ &\overset{l''=l'+2}{=} b_{l'+4} \\ &= \dots \\ &\overset{l''=l'+2}{=} b_{l'+4(r-s-1)+2} \\ &= b_{l''+4(r-s-1)+2} \\ &=: p \end{aligned}$$

as well as

$$\begin{aligned} b_{l'-4(r-s-1)-1} &= b_{l''-4(r-s-1)-1} \\ &\overset{l''=l'+2}{=} b_{l'-4(r-s-1)+1} \\ &= \dots \\ &\overset{l''=l'+2}{=} b_{l'-1} \\ &= b_{l''-1} \\ &\overset{l''=l'+2}{=} b_{l'+1} \\ &\overset{l''=l'+2}{=} b_{l'+3} \\ &= \dots \\ &\overset{l''=l'+2}{=} b_{l'+4(r-s-1)+1} \\ &= b_{l''+4(r-s-1)+1} \\ &=: q \end{aligned}$$

Furthermore we can assume that $p \neq q$ since otherwise we could re-choose l' and l'' such that we would be in case 2.2.1 again and we already solved that case.

All of this together implies

$$b[\underbrace{l'-4(r-s-1)-2}_{=j-1-4(r-s-1)-2}, \underbrace{l''+4(r-s-1)+2}_{=j+1+4(r-s-1)+2}] = (pq)^{2(r-s-1)+1}pqp(qp)^{2(r-s-1)+1}.$$

and hence

$$\begin{aligned} & b[j+3-4(r-s-1), j+3+4(r-s-1)] \\ = \;& (pq)^{2(r-s-1)}p(qp)^{2(r-s-1)} \\ = \;& b[l'-4(r-s-1), l'+4(r-s-1)] \\ = \;& a[k-4(r-s-1), k+4(r-s-1)]. \end{aligned}$$

And so Duplicator can set $\beta_{s+1}(x_2) = j+3$.

$$a: \quad \ldots \overbrace{pq\ldots pq}^{4(r-s-1)+2} \; p \underset{i=\alpha_{s+1}(x_1)}{\underset{\uparrow}{q}} \; p \overbrace{qp\ldots qp}^{4(r-s-1)+2} \ldots\ldots \overbrace{pq\ldots pq}^{4(r-s-1)+2} \underset{k=\alpha_{s+1}(x_2)}{\underset{\uparrow}{p}} \overbrace{qp\ldots qp}^{4(r-s-1)+2} \ldots$$

$$b: \quad \ldots \overbrace{pq\ldots pq}^{4(r-s-1)+2} \underset{l'}{\underset{\uparrow}{p}} \underset{j=\beta_{s+1}(x_1)}{\underset{\uparrow}{q}} \underset{l''}{\underset{\uparrow}{pq}} \underset{\beta_{s+1}(x_2)}{\underset{\uparrow}{p}} \overbrace{qp\ldots qp}^{4(r-s-1)}\ldots$$

□

7 Conclusion and Open Problems

Although we have seen two new characterizations for the languages, definable by first order formulas with only two different variables and succ as the only binary relation, one of these (the temporal logic one) seems to be only introduced to specifically fit this problem and so nothing new has been shown here. However, it is not clear a priori – although the actual proof is very easy – why it should even be possible to just invent another operator and then have the same kind of equivalence as for the classes including the $<$ relation.

A next step would be to investigate the hierarchy defined by the nesting depth of the **D** operator and show that it is strict – which it "should" be. This might be achieved by characterizing this hierarchy in terms of semigroup theory and then showing that the resulting hierarchy of pseudovarieties is strict. This approach was used by Thérien and Wilke [TW04] to prove that the Until-Since-hierarchy is strict.

Of course it should also be interesting to investigate the effects of allowing other combinations of temporal operators, e.g. only **XU** without **YS** (work has be done in this field already, e.g. in [TW96]) or unorthodox combinations like **XU** together with **O**.

Furthermore in the case without the $<$ relation it could be investigated what would happen if we allowed more than two different variables as in this case we would not automatically end up in the whole class of languages definable by temporal logic as we do with $<$, as seen in Theorem 3.1.

Bibliography

[DG08] V. Diekert and P. Gastin, *First-order definable languages*, Texts in Logic and Games, vol. 2, pp. 261–306, Amsterdam University Press, 2008.

[EFT94] H.-D. Ebbinghaus, J. Flum, and W. Thomas, *Mathematical logic*, 2nd ed., Springer Verlag, Berlin Heidelberg, 1994.

[Ehr61] A. Ehrenfeucht, *An application of games to the completeness problem for formalized theories*, Fundamenta Mathematicae **49** (1961), 129–141.

[EVW97] K. Etessami, M. Y. Vardi, and T. Wilke, *First-order logic with two variables and unary temporal logic*, Proceedings of the Twelfth Annual IEEE Symp. on Logic in Computer Science, LICS 1997 (Warsaw, Poland) (Glynn Winskel, ed.), IEEE Computer Society Press, June 1997, pp. 228–235.

[Fra54] R. Fraïssé, *Sur les classifications des systems de relations*, Ph.D. thesis, Paris, 1954.

[Imm99] N. Immerman, *Descriptive complexity*, Graduate Texts in Computer Science, Springer-Verlag, New York, 1999.

[Kam68] J. A. W. Kamp, *Tense logic and the theory of linear order*, Ph.D. thesis, University of California, Los Angeles, 1968.

[MP71] R. McNaughton and S. Papert, *Counter-free automata*, MIT Press, 1971.

[Pin86] J.-E. Pin, *Varieties of formal languages*, Foundations of Computer Science, Plenum Publishing Corporation, New York, 1986.

[Pnu77] A. Pnueli, *The temporal logic of programs*, IEEE Symposium on Foundations of Computer Science (FOCS), 1977, p. 46–57.

[Sch65] M. P. Schützenberger, *On finite monoids having only trivial subgroups*, Information & Control **8** (1965), 190–194.

[Str78] H. Straubing, *Varieties of recognizable sets whose syntactic monoids contain solvable groups*, Ph.D. thesis, University of California, Berkeley, 1978.

[Str94] ———, *Finite automata, formal logic, and circuit complexity*, Birkhäuser, Boston, MA, 1994.

[TW96] D. Thérien and T. Wilke, *Temporal logic and semidirect products: An effective characterization of the until hierarchy*, In Proceedings of the 37th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, 1996, pp. 256–263.

[TW98] ———, *Over words, two variables are as powerful as one quantifier alternation: $FO^2 = \Sigma_2 \cap \Pi_2$*, Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (Dallas, Texas), 24–26 May 1998, pp. 41–47.

[TW04] ———, *Nesting until and since in linear temporal logic*, Theory Comput. Syst. **37** (2004), no. 1, 111–131.