

Transfinite Context-Free Generative Grammars

Contents

1	Introduction	3
1.1	Related Work	3
1.2	Terminology and Notation	4
1.3	The Classical Theory	4
1.3.1	Generative Grammars	4
1.3.2	The Chomsky Hierarchy	5
1.3.3	Classical Results	6
2	Transfinite Generative Grammars	6
2.1	Motivation	6
2.2	Ordinal Languages	7
2.3	Taking Limits	7
3	Ordinal Type-2 Languages	9
3.1	Ordinal Context-Free Grammar	9
3.2	Ordinal Derivation Tree	10
3.3	Closure Properties	11
3.4	Examples	11
4	Ordinal Type-3 Languages	14
4.1	Ordinal Regular Grammar	15
4.2	Ordinal Finite State Automata	15
4.3	Examples	17
5	Properties of Ordinal Regular Languages	18
5.1	Pumping Lemma	18
5.2	Deflation Lemma	19
5.3	Size of Regular Languages	20
5.4	Closure Properties	21

1 Introduction

The aim of this essay is to generalise the lower part of the well-studied *Chomsky hierarchy*, used to classify generative grammars in computation theory, to a transfinite setting. In Section 2, we will define a variant of context-free grammar that generalises the classical notion to include computation on objects of general ordinal length. It turns out that there is a natural way of restricting these *ordinal context-free grammars* to transfinite notions of regular grammars with a corresponding machine model (Proposition 4.1). Unlike in the classical theory, when further introducing the corresponding notion of determinism for these *ordinal regular grammars*, we get a strictly weaker model of computation (Proposition 5.8). Despite this difference, we will see in Section 5 that some fundamental results from the classical theory can be adapted to the transfinite setting.

1.1 Related Work

The theory of transfinite computation is the area of computation theory studying the transfinite extensions of computation models (first introduced in [3]). The key idea is that we allow our notion of “algorithm” to involve an ordinal number of steps, beyond the positive integers as is traditional custom. Such a generalised notion of computation can thus perform infinitely many steps, and perform even more computation after it is done with all of them. A large part of the theory is concerned with extending classical models of computation (such as Turing machines or register machines) to the transfinite setting (see [4] for a systematic overview).

The existing transfinite computation literature is predominantly concerned with finding transfinite versions of comparatively strong models of computation and focuses mainly on the automata models. Examples of such generalised models include Infinite Time Turing (Register) Machines and Ordinal Turing (Register) Machines and many more (cf. [4, §2]). A few authors also define transfinite notions generalising regular languages via generalised finite state automata (cf. [4, §2.7.1], [7, §1]). Some work has also been done on investigating the notion of determinism in transfinite settings, examples of this include the discussions in [5, §4] and [6, §2.2].

1.2 Terminology and Notation

Throughout this essay, we will use the following standard terminology and notation from classical computation theory.

symbol		an atomic unit of computation
alphabet		a collection of symbols
word		a string of symbols (over some alphabet)
language		a collection of words (over some alphabet)
ε		the empty word/symbol

To better distinguish between variables representing symbols and words, we will denote the latter using boldface. While words will be denoted using lower-case letters, we will use upper-case letters for alphabets and languages. Both lower-case and upper-case letters will be used to denote symbols, depending on their nature. For any alphabet Σ , we will write Σ^* for the *Kleene closure* of Σ , denoting the language of all finite words over Σ . We extend this notation and also write L^* for the language of all finite concatenations of words from the language L . For languages L_1 and L_2 over alphabet Σ , we will also write $L_1 L_2$ for the languages of all concatenations of words from language L_1 with words from language L_2 and $\overline{L_1}$ for the *complement* $\Sigma^* \setminus L_1$ of L_1 . For a symbol or word x and an integer $k \geq 0$, we write x^k for the k -fold repetition of x .

1.3 The Classical Theory

1.3.1 Generative Grammars

Generative grammars give constructive descriptions of languages by specifying some replacement rules on an alphabet enriched with additional symbols only used in the generation process. A word is in the language generated by the grammar if it can be derived by a finite sequence of these replacements rules, starting with a specified initial symbol. In classical computation theory, we formally define generative grammars as follows (cf. [2, §1.2]).

Definition 1.1 (Grammar). A *grammar* G is a tuple (N, Σ, P, S) where N is a finite alphabet of *non-terminal symbols*, Σ is a finite alphabet of *terminal symbols* (disjoint from N), P is a finite set of *production rules* of the form

$$\mathbf{a} \rightarrow \mathbf{b},$$

with $\mathbf{a} \in N^* \setminus \{\varepsilon\}$ and $\mathbf{b} \in (N \cup \Sigma)^*$. Lastly, $S \in N$ is the *initial symbol*. The *language generated by G* , written $L(G)$, is the subset of Σ^* that can be obtained by a finite sequence of substring replacements according to the production rules, starting with the initial symbol S .

We can combine notation for multiple production rules with the same left-hand side, writing the shorthand

$$a \rightarrow b_1 \mid b_2 \mid \cdots \mid b_k \quad \text{for} \quad a \rightarrow b_1, a \rightarrow b_2, \dots, a \rightarrow b_k.$$

Some classical examples of languages given by generative grammars (with the initial symbol S) are given below.

N	Σ	P	Language
S	$(,)$	$S \rightarrow SS \mid \varepsilon \mid (S)$	Balanced strings of parentheses
S	a, b	$S \rightarrow aSb \mid \varepsilon$	$\{a^n b^n \mid n \geq 0\}$
S, X	a, b	$S \rightarrow aS \mid bX, X \rightarrow aS \mid bX \mid \varepsilon$	$\{wb \mid w \in \{a, b\}^*\}$

Table 1: Examples of generative grammars and the languages they generate

1.3.2 The Chomsky Hierarchy

Restricting the shape of the production rules for classical generative grammars gives rise to a hierarchy of languages called the *Chomsky hierarchy* (cf. [2, §11]). Each level in this hierarchy also has a corresponding automata model that generates the same class of languages. These machine models include *Turing Machines* (TM), *Linear Bounded Automata* (LBA), *Push-Down Automata* and *Finite State Automata* (FSA). We also give each level a name, which we use to refer to both the class of grammars and the class of languages generated by them.

Type	Production Rules	Name	Automaton
0	$u \rightarrow v \quad (u \neq \varepsilon)$	Computably Enumerable	TM
1	$uAw \rightarrow uvw$	Context-Sensitive	LBA
2	$A \rightarrow v$	Context-Free	PDA
3	$A \rightarrow aB \mid \varepsilon$	Regular	FSA

Table 2: The four levels of the Chomsky hierarchy

1.3.3 Classical Results

The following are some of the fundamental results from the classical theory concerning context-free and regular languages. They are all proved in detail in [2, §4.1, §4.3, §8] and [1, §4.1.1, §4.2.1, §7.3.2].

Proposition 1.1 (Closure Properties of Context-Free Languages). *If L_1, L_2 are two context-free languages over the same alphabet, then so are $L_1 \cup L_2$, $L_1 L_2$ and L_1^* . However, $L_1 \cap L_2$ and $\overline{L_1}$ are not necessarily context-free.*

Lemma 1.2 (Pumping Lemma). *For every regular language L there is a positive integer n with the following property: Every $\mathbf{w} \in L$ with $|\mathbf{w}| \geq n$ can be decomposed as $\mathbf{w} = \mathbf{w}_1 \mathbf{w}_2 \mathbf{w}_3$, where $|\mathbf{w}_1 \mathbf{w}_2| \leq n$, $\mathbf{w}_2 \neq \varepsilon$ and $\mathbf{w}_1 \mathbf{w}_2^k \mathbf{w}_3 \in L$ for every integer $k \geq 0$.*

Proposition 1.3 (Closure Properties of Regular Languages). *If L_1, L_2 are two regular languages over the same alphabet, then so are all of $\overline{L_1}$, $L_1 \cap L_2$, $L_1 \cup L_2$ and $L_1 L_2$.*

In sections 3.3 and 5 we will find equivalent statements in the transfinite setting for all the above results. We will then refer back to these for comparison.

2 Transfinite Generative Grammars

2.1 Motivation

Often, the definitions given in transfinite computation are not very canonical, with many seemingly arbitrary choices when defining the behaviour at limits. Having a transfinite notion of generative grammar and a corresponding Chomsky hierarchy could help unify some of the many variations and inform the choices made in the various definitions.

One of the big obstacles in transfinite computation theory is that the notion of non-determinism is more substantial than in the classical theory (cf. [5, §4], [6, §2.2]). Most classical models of computation do not gain any more power when non-determinism is introduced. This is rarely the case in transfinite computation, because an additional existential quantifier could range over collections of arbitrary cardinality (or even proper classes), as opposed to just a countable set. In Section 5, we will prove that this phenomenon occurs already at a transfinite equivalent of the type-3 level. This splitting of the Chomsky hierarchy is representative of an overarching dichotomy between determinism and non-determinism in transfinite computation theory. Studying these comparatively simple models of computation could shed light on how to deal with this obstacle in general.

2.2 Ordinal Languages

To extend the notion of generative grammar to the transfinite, we first need to define what we mean by a “word of transfinite length”. This definition is quite natural, and we adopt notation from the classical theory.

Definition 2.1 (Ordinal Word). Let Σ be a set and γ an ordinal. An *ordinal word* \mathbf{w} over the *alphabet* Σ of *length* γ is a function $\gamma \rightarrow \Sigma$. Let \mathbf{w}' be another ordinal word. We write $|\mathbf{w}|$ for the length of \mathbf{w} and ε for the unique *empty word* of length 0. Further, we write \mathbf{ww}' for the *concatenation* of \mathbf{w} and \mathbf{w}' of length $|\mathbf{w}| + |\mathbf{w}'|$ defined by

$$\mathbf{ww}'(\alpha) = \begin{cases} \mathbf{w}(\alpha), & \text{if } \alpha < |\mathbf{w}| \\ \mathbf{w}'(\beta), & \text{if } \alpha = |\mathbf{w}| + \beta \end{cases}$$

We also need a corresponding concept of a transfinite language. Slight care is needed here, as we also want to consider collections of ordinal words that are proper classes. We also define an ordinal version of the Kleene Star.

Definition 2.2 (Ordinal Language). An *ordinal language* L over alphabet Σ is a collection of ordinal words over Σ . Let \mathbf{w} be an ordinal word and L' be another ordinal language over Σ . We write Σ^* for the *ordinal Kleene closure* of Σ , the language of all ordinal words over Σ . Further, we write L^* for the language of ordinal length concatenations of words in L and denote by \bar{L} the ordinal language $\Sigma^* \setminus L$.

2.3 Taking Limits

The main task when seeking a meaningful definition of transfinite generative grammar is to decide what happens at limit stages. We need to decide on the result of applying a limit ordinal number of production rules. In the classical theory, consecutive applications of the production rules are usually represented in a linear fashion, connected by arrows (cf. [1, §5.1, §5.2]). The information about which part of the word the production rule is being applied to is usually omitted since even in the case where it is ambiguous, it does not impact the outcome of the derivation. The reason for this is the finite nature of these derivations. Let us consider the generative grammar G with initial symbol A , non-terminal alphabet $\{A, B\}$, terminal alphabet $\{a, b\}$ and production rules

$$A \rightarrow aA \mid aB \mid a, \quad B \rightarrow bA \mid bB.$$

This grammar generates the language of all finite words over $\{a, b\}$ that start and end in the symbol a . In the classical theory, the derivation of the word $abaaba$ could be written

$$A \rightarrow aB \rightarrow abA \rightarrow abaA \rightarrow abaaB \rightarrow abaabA \rightarrow abaaba.$$

Now suppose that we wish to apply the production rule $A \rightarrow aA$ a total of ω times to the initial symbol A . We need to choose the result of the partial derivation

$$A \rightarrow aA \rightarrow aaA \rightarrow aaaA \rightarrow aaaaA \rightarrow \dots,$$

where the dots represent an ω limit of production rule applications. The only natural choice is to have this sequence result in the word $a^\omega A$, so we seek a notion of limit that behaves in this way. On the other hand, if we alternate production rules $A \rightarrow aB$ and $B \rightarrow bA$, it is not clear what the result of the derivation

$$A \rightarrow aB \rightarrow abA \rightarrow abaB \rightarrow ababA \rightarrow \dots$$

should be. We want it to be of the form $(ab)^\omega X$ where X is a non-terminal that represents, in some sense, the limit of the non-terminals A and B . Since we do not want to introduce new symbols, we can instead fix a total order on the set of our non-terminals of G and then set $S = \max \{A, B\}$.

Now let us look at a more troublesome scenario with more non-terminals involved. Consider the grammar G with unique non-terminal S , terminal alphabet $\{a, b\}$ and production rules

$$S \rightarrow SS \mid Sa \mid Sb \mid \varepsilon.$$

Now the derivation $S \rightarrow SS \rightarrow SSS \rightarrow SSSS \rightarrow \dots$ is ambiguous, since it is not clear which of the symbols the production rule was applied. In the classical theory, this is not an issue, since all possible derivations yield the same result. In the above infinite derivation, however, it does make a difference. If the symbol replaced is always the last one, we would naturally want the limiting word to be $S^\omega S$, in accordance with the natural limits we've seen before. If however, we eventually always replaced the second to last symbol, we would pick the limit $S^\omega SS$ instead. With the grammar defined above we do run into another problem as well. Consider the derivation chain

$$S \rightarrow Sa \rightarrow Sba \rightarrow Saba \rightarrow Sbaba \rightarrow \dots,$$

where we alternate between the production rules $S \rightarrow Sa$ and $S \rightarrow Sb$. In this case, while there is an obvious pattern in the word of terminals generated, it is not possible to make a sensible choice for what should be at the second position of the resulting word. We could overcome this issue by again defining some limiting x of the terminals a, b and then choosing the result of the derivation to be Sx^ω . This however is a very intrusive operation that destroys the alternating nature of the derivation leading up to it, so not a desirable solution.

The above examples suggest that these linear representations of derivations used in the classical theory do not carry sufficient information to be naturally extended to the transfinite setting. Furthermore, being able to have unlimited derivations that “generate symbols to their right” seems to cause problems at limit stages. This apparent asymmetry between left and right is not unexpected, given that increasing and decreasing sequences behave very differently

in the ordinals. The upshot of this section is that to get a coherent notion of generative grammar in the transfinite setting, we have to define derivations on a richer structure that takes into account the above observations. In the case of context-free grammars, derivations can also be represented using derivation trees. It turns out that this is a much more natural way to work with transfinite derivations.

3 Ordinal Type-2 Languages

This section aims to extend the notion of context-free (type-2) languages to the transfinite setting. We first extend the definition of context-free grammars, then we define what it means for such a grammar to generate an ordinal word.

3.1 Ordinal Context-Free Grammar

While the classical definition of generative grammar is rather canonical, there are many more choices to be made in the transfinite setting. Since we work over the entire class of ordinals, the sizes of the various sets in the definition could be limited to arbitrary cardinalities. The same goes for the word lengths in the production rules. For the sake of scope and simplicity, we will only consider grammars that are **finite**. This means all sets and words in the specification of the grammar are finite. With this restriction, the definition of ordinal context-free grammar looks almost like the classical definition. The only difference is that we additionally require the set of non-terminals to be totally ordered, which will become important in the limit steps of the derivation process defined in section 3.2.

Definition 3.1 (Ordinal Context-Free Grammar). An *ordinal context-free grammar* (or *OCFG* for short) G is a tuple (N, Σ, P, S) where N is a finite, **totally ordered** set of *non-terminal symbols*, Σ is a finite set of *terminal symbols*, disjoint from N , P is a finite set of *production rules* of the form

$$A \rightarrow \mathbf{a},$$

where $A \in N$, $\mathbf{a} \in (N \cup \Sigma)^*$ and $S \in N$ is the *initial symbol*.

The transfinite aspect of these grammars is only present in the ability to apply a transfinite number of production rules in the derivation process. The benefit of requiring our ordinal grammars to be finitely describable is that they compare very directly to classical grammars. The fact that they can be finitely encoded also allows us to consider classical decision problems for these grammars.

3.2 Ordinal Derivation Tree

As discussed in Section 2.3, the linear approach is not able to capture the much richer structure of transfinite-length derivations. Thus we define a transfinite notion of a derivation tree with arbitrary ordinal depth. Since such a tree might have vertices at limit depths, we cannot define it using a parent-child relation. Rather, we have to use a global ancestry relationship.

Definition 3.2 (Ordinal Depth Tree). An *ordinal depth tree* T is a tuple $(X, 0, <, \triangleleft)$ where X is a set of *vertices*, 0 is an element of X called the *root*. Also, $<$ is a partial order on X called the *ancestry relation* with least elements 0 , the property that for every $x \in X$, the set $\{y \in X \mid y < x\}$ is well-ordered and such that every chain is bounded above. Further, \triangleleft is a total order on X that induces a well-order on the maximal chains of $(X, <)$ when ordered by their first deviation. For vertices $x, y \in X$, we say that x covers y if $x > y$ and there is no $z \in S$ with $x > z > y$ (this recovers the weaker child-parent relation). We call the maximal vertices of T the *leaves* of T .

The last two conditions on the order $<$ guarantee that every vertex has a unique ordinal depth and every branch of the tree ends in a leaf vertex, rather than in a limit of vertices. The only purpose of the secondary relation on X is to well-order the descendent of each vertex and thus induce a well-order on the leaves of the ordinal depth tree.

With the structure part of the derivation tree defined, we now decorate the tree with the symbols of our grammar. Just like in the classical case, leaf vertices are decorated with terminal symbols (or the empty symbol) while non-leaf vertices are assigned non-terminals. Of course, we require this assignment to be in accordance with the production rules of our ordinal grammar, with limit vertices determined by the limsup of the non-terminals above it.

Definition 3.3 (Ordinal Derivation Tree). Given a OCFG $G = (N, \Sigma, P, S)$, an *ordinal derivation tree* of G is an ordinal depth tree $T = (X, 0, <, \triangleleft)$ together with a map $f : T \rightarrow N \cup \Sigma \cup \{\varepsilon\}$ such that $f(0) = S$, $f(x) \in \Sigma \cup \{\varepsilon\}$ for each leaf x of T and $f(x) = \limsup \{f(y) \mid y < x\}$ for limit vertices $x \in X \setminus 0$. Furthermore, if $y_1 \triangleleft y_2 \triangleleft y_3 \triangleleft \dots$ are the vertices covering x , then

$$f(x) \rightarrow f(y_1)f(y_2)f(y_3)\dots$$

must be a production rule of G . We say that the derivation tree of G *derives* the ordinal word obtained by concatenating the values of f at the leaves of T , in the order induced by \triangleleft . In doing so, the symbol ε is treated as the empty symbol, not contributing to the generated word. We say that the grammar G derives the ordinal word w if there is a derivation tree of G that derived w .

Note that the above definitions are phrased in a way that also allows production rules with infinite words on the right-hand side, which we will not study in this essay. From here we conclude with the definition of an ordinal context-free language.

Definition 3.4 (Ordinal Context-Free Language). An ordinal language L is called an *ordinal context-free language* (or *OCFL* for short) if there is an OCFG G that derives exactly the words in L . In this case, we say that L is the language of the grammar G .

3.3 Closure Properties

Our notion of ordinal context-free language resembles the classical notion not only in a very similar definition but also behaves very similarly. In particular, these languages satisfy the same closure properties laid out in section 1.3.3 (unions, concatenations and Kleene closure). Additionally, OCFLs are closed under ordinal Kleene closure.

Proposition 3.1 (Closure Properties of OCFLs). *If L_1 and L_2 are two OCFLs over the same alphabet Σ , then so are all of*

$$L_1 L_2, L_1 \cup L_2, L_1^*, L_1^{\otimes}.$$

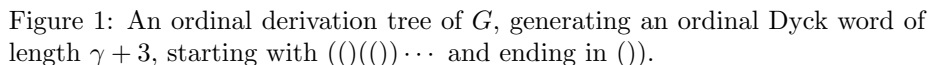
Proof. Let $G_1 = (N_1, \Sigma, P_1, S_1)$ and $G_2 = (N_2, \Sigma, P_2, S_2)$ be OCFGs that generate L_1 and L_2 respectively (with N_1 and N_2 disjoint). Define a new OCFG $G = (N, \Sigma, P, S)$ with $N = N_1 \sqcup N_2 \sqcup \{S\}$ and $P = P_1 \sqcup P_2$. If we add the production rule $S \rightarrow S_1 S_2$ to P , we have $L(G) = L_1 L_2$ and if we instead add $S \rightarrow S_1 \mid S_2$, we get $L(G) = L_1 \cup L_2$. Now consider the grammar G with additional production rule $S \rightarrow S_1 S \mid \varepsilon$. This rule alone generates ordinal words of repeated symbol S_1 . Thus, we get $L(G) = L_1^{\otimes}$. If instead we add the production rule $S \rightarrow S S_1 \mid \varepsilon$ we get $L(G) = L_1^*$, since S produces symbols to its right and hence can only do so finitely many times. Thus we can find an OCFG that generates each of the languages in question, making them all OCFLs. \square

The above proof suggests that there is a duality between finite and arbitrary ordinal length. This phenomenon is closely linked to the fact that in the ordinals any decreasing sequence must be finite, while increasing sequences can be of arbitrary length.

3.4 Examples

The Ordinal Dyck Language

Consider the ordinal context-free grammar G with single non-terminal S , terminal symbols $\{(\,,\,)\}$ and production rules $S \rightarrow SS \mid (S) \mid \varepsilon$. In the classical theory this grammar is called the *Dyck Grammar* and the language it generates is the set of all finite words of balanced parentheses. In the transfinite context, we also include ordinal words that satisfy a generalised notion of “being balanced”. We call this larger language the *Ordinal Dyck Language*. Consider for example the following ordinal derivation tree of G , where the dashed section represents a limit of length γ .



We note that a typical ordinal depth tree grows into the bottom-right direction just like the above tree. This is because the well-order condition on the leaves of the tree implies that all chains can only have finitely many offshoots to the right. Thus, any infinite chain is eventually the right-most possible path through the tree, with all offshoots to the left.

$$f(0) = f(|\mathbf{w}|) = 0,$$

$$f(\alpha + 1) = \begin{cases} f(\alpha) + 1, & \text{if } \mathbf{w}(\alpha + 1) = (\\ f(\alpha) - 1, & \text{if } \mathbf{w}(\alpha + 1) =) \end{cases} \quad \text{for successors } \alpha + 1 \leq |\mathbf{w}|,$$

$$f(\gamma) = \liminf \{f(\alpha) : \alpha < \gamma\} \quad \text{for non-zero limits } \gamma \leq |w|.$$

12

Proof. If $|\mathbf{w}| < 2$ the result is trivial. We induct on the length of \mathbf{w} . Let $\mathbf{w} \in \{(\,,\,)\}^*$ and $f : |\mathbf{w}| + 1 \rightarrow \omega$ be a function with the above properties. If there is no ordinal $0 < \alpha < |\mathbf{w}|$ with $f(\alpha) = 0$, not that \mathbf{w} must start in (and end in). Let \mathbf{w}' be the word obtained by removing the first and last symbol of \mathbf{w} . Now $|\mathbf{w}'| < |\mathbf{w}|$ and after subtracting 1, f restricts to a function with the desired property on $|\mathbf{w}'| + 1$. Thus \mathbf{w}' is in the Dyck language by induction and $\mathbf{w} = (\mathbf{w}')$ is also, due to the production rule $S \rightarrow (S)$.

If instead there is some $0 < \alpha < |\mathbf{w}|$ with $f(\alpha) = 0$, then if there is a largest such α we write $\mathbf{w} = \mathbf{w}_1\mathbf{w}_2$ with $|\mathbf{w}_1| = \alpha < |\mathbf{w}|$. Since f is strictly positive on \mathbf{w}_2 , we have $\mathbf{w} = \mathbf{w}_1(\mathbf{w}_2')$ with \mathbf{w}' in the Dyck Language by a similar argument to the one above. Hence using the production rules $S \rightarrow SS$ and $S \rightarrow (S)$ we find that \mathbf{w} is also in the Dyck language. If there is no largest such α , we can find ω many of them $\alpha_1, \alpha_2, \dots$ and write $\mathbf{w} = \mathbf{w}_1\mathbf{w}_2 \dots$ with $|\mathbf{w}_1\mathbf{w}_2 \dots \mathbf{w}_k| = \alpha_k$ for all integers $k \geq 1$. Restricting f to these prefixes we find that \mathbf{w}_k is in the Dyck language for all $k \geq 1$ using the inductive hypothesis. Using the production rule $S \rightarrow SS$ a total of ω times we can derive the word S^ω , and substituting the derivation tree for the \mathbf{w}_k for these symbols we find that \mathbf{w} is also in the Dyck language.

Conversely, let \mathbf{w} be a word in the Dyck language and fix an ordinal derivation tree that generates \mathbf{w} . For any ordinal $\alpha < |\mathbf{w}|$ consider the leaf vertex of the derivation tree that corresponds to $\mathbf{w}(\alpha)$. Look at the unique path from the root to this vertex. We now set $f(\alpha)$ to be the number of vertices on this path corresponding to the production rule $S \rightarrow (S)$, where the path either continued along the first or second branch. Since this rule has a third (non-trivial) offshoot, there can only be finitely many such vertices on the path. Also set $f(|\mathbf{w}|) = 0$. It follows from the structure of the ordinal derivation tree and the nature of the production rules that f satisfies the desired properties. \square

Alternating Finite Runs

Consider the ordinal context-free grammar G with non-terminals $\{S > A > B\}$, terminals $\{a, b\}$ and production rules

$$S \rightarrow ABS \mid A \mid \varepsilon, \quad A \rightarrow Aa \mid a, \quad B \rightarrow Bb \mid b.$$

Illustrated below is part of a possible derivation tree of G where the dashed section again represents a limit stage of length γ .

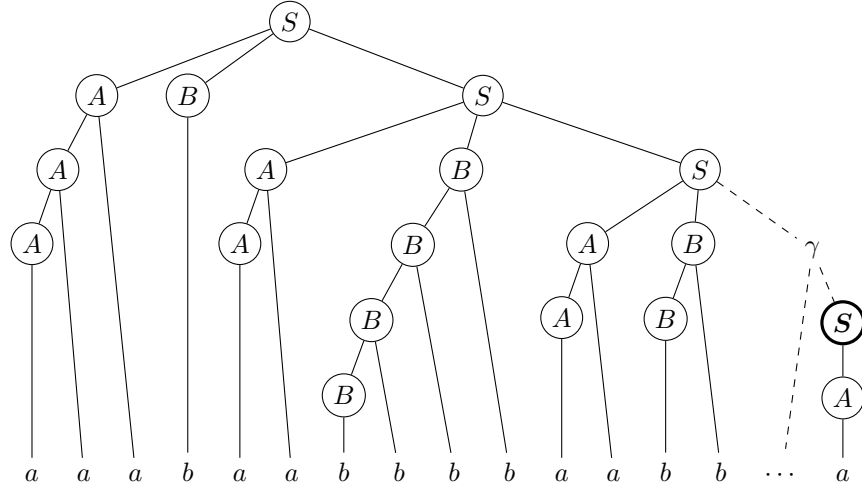


Figure 2: An ordinal derivation tree of G , generating a word of length $\gamma + 1$ starting with $aaabaabbbbaabb\dots$ and ending in the symbol a .

The single production rule involving the non-terminal S produces ordinal words of alternating symbols A and B , where all symbols at limit positions are A . This is because the order on the non-terminals introduces a symbol S at every limit stage, which can only be removed by replacing it with the symbol A . Since the production rules for A and B produce terminals to their right, they each generate finite, non-empty words in only the terminal symbol a and b respectively. Putting everything together, we find that $L(G)$ is the language of ordinal words over alphabet $\{a, b\}$ that do not contain infinite runs of a single symbol, and have symbol a at all limit positions (including the start of the word).

4 Ordinal Type-3 Languages

In the classical theory, the production rules of context-free (type-2) grammars can be further restricted to regular (type-3) grammars as described in section 1.3.2 on the Chomsky hierarchy. It could be expected that a similar restriction is possible for our notion of ordinal context-free grammar. While this is still the case, the transfinite theory diverges from the classical theory by offering two distinct possible restrictions. We will wait until Section 5 to prove that these two restrictions are not equivalent.

4.1 Ordinal Regular Grammar

Definition 4.1 (Ordinal Regular Grammar). An ordinal context-free grammar is called an *ordinal regular grammar* (or *ORG* for short) if all its production rules are of the form

$$A \rightarrow aB \quad \text{or} \quad A \rightarrow \varepsilon,$$

where $A, B \in N$, $a \in \Sigma$. If in addition for every $A \in N$, $a \in \Sigma$ there is at most one $B \in N$ with $A \rightarrow aB \in P$, then we call the grammar *deterministic* (or a *DORG* for short).

Definition 4.2 (Ordinal Regular Language). An ordinal language L is called a (*deterministic*) *ordinal regular language* (or (*D*)*ORL* for short) if there is a (*D*)*ORG* G that derives exactly the words in L . In this case, we say that L is the language of grammar G .

In the classical theory, the distinction between deterministic and non-deterministic grammars is only superficial. The two definitions are easily shown to be equivalent and thus there is a canonical way of defining type-3 languages. At the end of Section 5, we will see that this is not the case in the transfinite setting, where deterministic ordinal regular grammars are strictly weaker.

4.2 Ordinal Finite State Automata

As in the classical theory, regular languages are most easily reasoned about from within a machine model (cf. [1, §2.2, §2.3]). Luckily the translation between the grammar and automaton is very similar in the transfinite setting.

Definition 4.3 (Ordinal Finite State Automata). An *ordinal finite state automaton* (or *OFSA* for short) M is a tuple $M = (N, \Sigma, \delta, S, F)$ where N is a totally ordered, finite set of *states*, Σ is a finite alphabet, δ is a function $N \times \Sigma \rightarrow \mathcal{P}(N)$, $S \in N$ is the *initial state* and $F \subseteq N$ is a subset of *accepting states*. For an ordinal word $\mathbf{w} \in \Sigma^{\otimes}$ we further define a *path* of \mathbf{w} through M to be a function $f : |\mathbf{w}| + 1 \rightarrow N$ such that

$$\begin{aligned} f(0) &= S, \\ f(\alpha + 1) &= \delta(f(\alpha), \mathbf{w}(\alpha)) && \text{for successors } \alpha + 1 \leq |\mathbf{w}|, \\ f(\gamma) &= \limsup \{f(\alpha) \mid \alpha < \gamma\} && \text{for limits } \gamma \leq |\mathbf{w}|. \end{aligned}$$

If in addition $f(|\mathbf{w}|) \in F$, we call the path f *accepting*. An ordinal word \mathbf{w} over Σ is said to be *accepted* by the OFSA M if there is an accepting path for \mathbf{w} . The *language accepted by* M is the language of all ordinal words accepted by M . If in addition, $|\delta(A, a)| \leq 1$ for all $A \in N$ and $a \in \Sigma$, then we call the OFSA *deterministic* (or an *DOFSA* for short).

In the classical theory, the above is usually done with the notion of an *extended transition function* $\Delta : N \times \Sigma^* \rightarrow \mathcal{P}(N)$. While this method could be adapted for deterministic ordinal regular automata, such a function does not carry enough information to define the behaviour in the non-deterministic case. This is because at limit stages we need to know the exact path already taken through the automaton, rather than just the possible states at every step in the past. This surplus in necessary information is also the intuitive reason why non-determinism is a much stronger notion in the transfinite setting.

Proposition 4.1. *An ordinal language is generated by some (D)ORG if and only if it is accepted by some (D)OFSA.*

Proof. Let $G = (N, \Sigma, P_G, S)$ be an ORG. Now we define the OFSA $M = (N, \Sigma, \delta_M, S, F_M)$ by setting

$$\delta(A, a) = \{B \in N_G \mid A \rightarrow aB \in P_G\}$$

and

$$F_M = \{A \in N_G \mid A \rightarrow \varepsilon \in P_G\}.$$

It follows straight from the definitions that $L(M) = L(G)$. Conversely, given an OFSA $M = (N, \Sigma, \delta_M, S, F_M)$, define the ORG $G = (N, \Sigma, P_G, S)$ by setting

$$P_G = \{A \rightarrow \varepsilon \mid A \in F_M\} \cup \{A \rightarrow aB \mid A \in N_M, B \in \delta(A, a)\}.$$

It again follows straight from the definitions that $L(G) = L(M)$. Also, note that both directions preserve the condition for the automaton or the grammar to be deterministic. \square

As alluded to before, we will prove in Section 5 that unlike in the classical theory, the notions of DOFSA and OFSA are not equivalent. The classical proof of this equivalence makes use of what is called the *subset construction* (cf. [2, §2.3]). This line of reasoning crucially uses the fact that the power set of a finite set is finite. For the same reason mentioned above concerning the extended transition function, this argument does not work in the transfinite setting. While the machine itself is finite, infinite information on the past is required to determine what state to go to at a limit step.

In our definition of deterministic ordinal regular automata, we allow the transition function to map to the empty set of states, which allows for simpler diagrams. However, the following lemma shows that the machine does not lose any power if we demand the transition function to only map to singletons, a property which we will need in Section 5.4.

Lemma 4.2. *For every DOFSA $M = (N, F, \Sigma, \delta, S)$ there is another DOFSA $M' = (N', F, \Sigma, \delta', S)$ that accepts the same language as M and has $|\delta'(A, a)| = 1$ for all $A \in N'$ and $a \in \Sigma$.*

Proof. We define M' by setting $N' = N \sqcup \{D\}$ and $\delta' : N' \times \Sigma \rightarrow N'$ where

$$\delta'(A, a) = \begin{cases} \delta(A, a) & \text{if } |\delta(A, a)| = 1 \\ \{D\} & \text{if } |\delta(A, a)| = 0 \end{cases}$$

for all $A \in N$ and $a \in \Sigma$ and $\delta'(D, a) = \{D\}$ for all $a \in \Sigma$. By construction $|\delta'(A, a)| = 1$ for all $A \in N', a \in \Sigma$ and it is easy to check that $L(M') = L(M)$. \square

Our choice to restrict grammars to finite sets means that the above notion of transfinite finite state automaton is conceptually antipodal to the notion defined in [4, §2.7.1], where the grammar itself is too large to be modelled by a set. Our notion also crucially deviates from the various *omega-automata* defined in [7, §1] which only operate on words of length $\leq \omega$ and thus give a strictly weaker model.

4.3 Examples

Constant Suffix Language

Consider the (deterministic) ordinal regular grammar G with non-terminals $\{S > X\}$, terminals $\{a, b\}$ and production rules

$$S \rightarrow aS \mid bX, \quad X \rightarrow aS \mid bX \mid \varepsilon.$$

Illustrated below is the (deterministic) ordinal FSA corresponding to G .

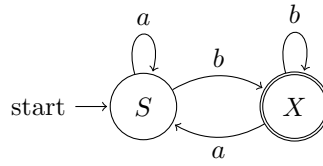


Figure 3: The DOFSA corresponding to the DORG G .

The language generated by G is easily seen to be the language of ordinal words over the alphabet $\{a, b\}$ that either end in the symbol b or whose tail is a limit containing only the symbol b .

Alternating Language with Priority

Consider the (deterministic) ordinal context-free grammar G with non-terminals $\{S < A < B\}$, terminals $\{a, b\}$ and production rules

$$S \rightarrow aB \mid bA \mid \varepsilon, \quad A \rightarrow aB \mid \varepsilon, \quad B \rightarrow bA \mid \varepsilon.$$

Illustrated below is its corresponding (deterministic) ordinal finite state automaton with the added dead state D described in the previous lemma.

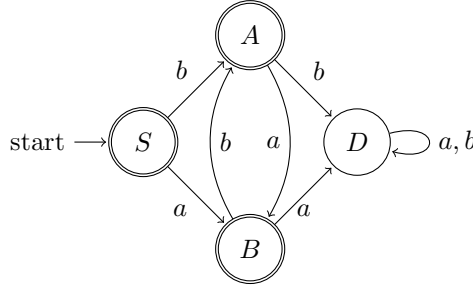


Figure 4: The DOFSA corresponding to the ORG G , with added dead state D .

It is not hard to see that $L(G)$ is the language of all ordinal words that alternate between symbols a and b , with no symbol a at any non-zero limit position. The latter condition is due to the choice $A < B$, sending the machine to state B at every limit stage.

5 Properties of Ordinal Regular Languages

5.1 Pumping Lemma

The main tool for arguing about the structure of regular languages in the classical theory is the pumping lemma (cf. [1, §4.1]). Considering the similarities between the classical and the transfinite automata model, it is easy to see that the statement of the classical pumping lemma still applies to ordinal regular languages. However, we would like to pump not only an integer number but rather an arbitrary ordinal number of times.

Lemma 5.1 (Ordinal Pumping Lemma). *For every ORL L there is a positive integer n with the following property: Every $w \in L$ with $|w| \geq n$ can be decomposed as $w = w_1 w_2 w_3 w_4$, where $|w_1 w_2 w_3| \leq n$, $w_2 w_3 \neq \varepsilon$ and $w_1 w_2 (w_3 w_2)^\alpha w_3 w_4 \in L$ for every ordinal α .*

When trying to adapt the classical proof, the only obstacle is pumping through limit stages where we have to be careful not to violate the limsup condition. This can be overcome by first creating a substring that can be pumped, whose leading symbol is maximal in the total order.

Proof. Let $M = (N, F, \Sigma, \delta, S)$ be an OFSA that accepts the language L . We will show that $n = |N|$ satisfies the desired condition. Let $\mathbf{w} \in L$ be such that $|\mathbf{w}| \geq n$ and let f be an accepting path of \mathbf{w} through M . Using the pigeonhole principle, there are integers $0 \leq t_1 \leq t_2 < t_3 \leq n$ such that $f(t_1) = f(t_3)$ and $f(t_2) = \max\{f(t_1), f(t_1 + 1), \dots, f(t_3)\}$. We decompose $\mathbf{w} = \mathbf{w}_1\mathbf{w}_2\mathbf{w}_3\mathbf{w}_4$ such that $|\mathbf{w}_1| = t_1$, $|\mathbf{w}_1\mathbf{w}_2| = t_2$ and $|\mathbf{w}_1\mathbf{w}_2\mathbf{w}_3| = t_3 \leq n$. Note also that $|\mathbf{w}_2\mathbf{w}_3| = t_3 - t_1 > 0$. Now for any ordinal α , let $\mathbf{w}_\alpha = \mathbf{w}_1\mathbf{w}_2(\mathbf{w}_3\mathbf{w}_2)^\alpha\mathbf{w}_3\mathbf{w}_4$ and define $f_\alpha : |\mathbf{w}_\alpha| + 1 \rightarrow N$ by

$$\begin{aligned} f_\alpha(k) &= f(k) && \text{for } 0 \leq k \leq |\mathbf{w}_1\mathbf{w}_2|, \\ f_\alpha(t_2 + (t_3 - t_1)\beta + k) &= f(t_2 + k) && \text{for } 0 \leq k < |\mathbf{w}_3| \text{ and } \beta < \alpha, \\ f_\alpha(t_2 + (t_3 - t_1)\beta + k) &= f(t_1 + k) && \text{for } |\mathbf{w}_3| \leq k < |\mathbf{w}_2\mathbf{w}_3| \text{ and } \beta < \alpha, \\ f_\alpha(t_2 + (t_3 - t_1)\alpha + \beta) &= f(t_2 + \beta) && \text{for } \beta \leq |\mathbf{w}_3\mathbf{w}_4|. \end{aligned}$$

It follows from the nature of the decomposition that f_α is an accepting path of \mathbf{w}_α through M . Hence, $\mathbf{w}_\alpha \in L$ for all ordinals α . \square

5.2 Deflation Lemma

In the classical theory, the pumping lemma can also be used to pump down words (by using $k = 0$). While we can also do this in the transfinite setting, the classical idea only removes finitely many symbols at a time, making it impossible to shrink below limit stages. We thus need another result allowing us to remove “large” parts of a word.

Lemma 5.2 (Deflation Lemma). *For every ORL L and any $\mathbf{w} \in L$ with $|\mathbf{w}| \geq \omega$, we can write $\mathbf{w} = \mathbf{w}_1\mathbf{w}_2\mathbf{w}_3$ where $|\mathbf{w}_1\mathbf{w}_3| < |\mathbf{w}|$ and $\mathbf{w}_1\mathbf{w}_3 \in L$. Additionally, we can pick $|\mathbf{w}_1|$ arbitrarily close to the largest limit below $|\mathbf{w}|$.*

Proof. Let $M = (N, F, \Sigma, \delta, S)$ be an OFSA that accepts the language L . Let $\mathbf{w} \in L$ and let f be an accepting path of \mathbf{w} through M . Set $|\mathbf{w}| = \gamma + n$ where $\gamma > 0$ is a limit, $n \geq 0$ an integer. Since N is finite and $f(\gamma) = \limsup\{f(\alpha) \mid \alpha < \gamma\}$, we must have $f(\alpha) = f(\gamma)$ for some $\alpha < \gamma$. Write $\mathbf{w} = \mathbf{w}_1\mathbf{w}_2\mathbf{w}_3$ where $|\mathbf{w}_1| = \alpha$ and $|\mathbf{w}_1\mathbf{w}_2| = \gamma$. Now define $f' : |\mathbf{w}_1\mathbf{w}_3| + 1 \rightarrow N$ by setting

$$\begin{aligned} f'(\beta) &= f(\beta) && \text{for all ordinals } \beta \leq \alpha, \\ f'(\alpha + k) &= f(\gamma + k) && \text{for all integers } 0 \leq k \leq n. \end{aligned}$$

By the choice of α it follows that f' is an accepting path of $\mathbf{w}_1\mathbf{w}_3$ through M . Hence, $\mathbf{w}_1\mathbf{w}_3 \in L$ and also $|\mathbf{w}_1\mathbf{w}_3| < \gamma \leq |\mathbf{w}|$. Picking α arbitrarily large below γ gives the additional requirement. \square

Note that the above proof makes crucial use of the fact that the alphabet of non-terminals is finite, and the particular choice of our limiting operation.

5.3 Size of Regular Languages

The results from the last section are crucial in understanding not only the structure but also the size of ordinal regular languages. The following lemma is concerned with the structure of the ordinal regular language around the first limit stage ω .

Lemma 5.3. *An ORL L contains an infinite word if and only if it contains arbitrarily long finite words.*

We will shortly need this result to show that any non-empty ORL L contains a finite word of arbitrary length.

Proof. Let L be an ORL. If L contains arbitrarily long finite words, then it must contain a word of length at least n , where n is as in the pumping lemma. In particular, we can apply the pumping lemma to pump up the word ω times, resulting in an infinite word of the language. Conversely, if L contains an infinite word, let n be an arbitrarily large integer and let $\mathbf{w} \in L$ be a shortest word of length at least n . If $|\mathbf{w}| \geq \omega$, we find a shorter word in L still longer than n , using the deflation lemma, contradiction minimality of $|\mathbf{w}|$. It follows that $n \leq |\mathbf{w}| < \omega$ and since n was arbitrary, the result follows. \square

We can now prove the main result of this section. In the classical theory, every regular language is either finite or the lengths of its words eventually have a linear structure in ω and thus have positive density. The following proposition gives an analogous dichotomy for ordinal regular languages.

Proposition 5.4. *Every ORL L is either finite or the lengths of its words are unbounded below every non-zero limit ordinal.*

It follows in particular that if an ORL L is infinite, then it is a proper class containing words of every cardinal length.

Proof. Let L be an ORL. If L is infinite, then since Σ is finite, L must contain arbitrarily long words. By the previous lemma, L must therefore contain arbitrarily long finite words. In particular, L contains a word \mathbf{w} with $n \leq |\mathbf{w}| < \omega$ where n is as in the pumping lemma. Thus, we can decompose $\mathbf{w} = \mathbf{w}_1\mathbf{w}_2\mathbf{w}_3\mathbf{w}_4$ such that $|\mathbf{w}_1\mathbf{w}_2\mathbf{w}_3| \leq n$, $\mathbf{w}_2\mathbf{w}_3 \neq \varepsilon$ and $\mathbf{w}_\alpha = \mathbf{w}_1\mathbf{w}_2(\mathbf{w}_3\mathbf{w}_2)^\alpha\mathbf{w}_3\mathbf{w}_4 \in L$ for all ordinals α . Let $0 < k = |\mathbf{w}_3\mathbf{w}_2| < \omega$. Since

$$k\alpha \leq |\mathbf{w}_\alpha| = |\mathbf{w}_1\mathbf{w}_2| + |\mathbf{w}_3\mathbf{w}_2|\alpha + |\mathbf{w}_3\mathbf{w}_4| < n + k\alpha + \omega,$$

we conclude that as α varies, $|\mathbf{w}_\alpha|$ is unbounded below every limit ordinal. Thus, the result follows. \square

5.4 Closure Properties

In this section, we will investigate the closure properties of ordinal regular languages. In the classical theory, regular languages are closed under the natural operations of unions, intersections, concatenations and complementation (as seen in Section 1.3.3). However, things are more involved in the transfinite setting. In particular, we will see that with regard to closure properties, deterministic ORLs behave almost complementary to general ORLs. The following observation is crucial.

Lemma 5.5. *There exist DORLs L_1 and L_2 such that $L_1 \cap L_2$ is not an ORL.*

Proof. Let L_1 be the DORL generated by the DORG studied in the latter example from section 4.3. There we have seen that L_1 is the language of alternating symbols a and b with no symbol b at a non-zero limit position. Now define L_2 similarly, but with the roles of the symbols a and b swapped. Considering the symbol at position ω , we note that L_1 and L_2 have no words of infinite length in common. However, all finite words of alternating symbols a and b are in both L_1 and L_2 . We conclude that $|L_1 \cap L_2| = \aleph_0$. Comparing this with the dichotomy result on the sizes of ORL, this implies that $L_1 \cap L_2$ is not an ORL. \square

The above phenomenon is unlike what happens in the classical theory, as classical regular languages (no matter deterministic or not) are closed under intersection. One way of proving this directly is to build what is known as the *product automaton* of two regular automata, which operates on the Cartesian product of the two sets of states. In the classical case, this automaton can fully capture the behaviour of both automata simultaneously. In the transfinite setting, however, the limit stages again form an obstacle. In particular, there is no total order of the Cartesian product for which the limsup operation commutes with the projection maps.

Proposition 5.6 (Closure properties of DORLs). *If L_1, L_2 are DORLs over the same alphabet, then so is $\overline{L_1}$. However, $L_1 \cap L_2$ and $L_1 \cup L_2$ are not necessarily DORLs.*

Proof. Let $M = (N, F, \Sigma, \delta, S)$ be a DOFSA that accepts language L . Using one of the previous lemmas, we can assume that $|\delta(A, a)| = 1$ for all $A \in N, a \in \Sigma$. Now define a new DOFSA $M' = (N, F', \Sigma, \delta, S)$ where $F' = N \setminus F$. Using the fact that every word has a unique path through M , it follows that a word is accepted by M' if and only if it is not accepted by M . Thus $L(M') = \overline{L(M)}$ and we conclude that \overline{L} is a DORL. The negative results follow from the previous lemma and De Morgan's law, together with the fact that every DORL is an ORL. \square

The argument that DORLs are closed under complementation is the same as in the classical theory.

Proposition 5.7 (Closure properties of ORLs). *If L_1, L_2 are ORLs over the same alphabet, then so is $L_1 \sqcup L_2$. However, $L_1 \cap L_2$ and $\overline{L_1}$ are not necessarily ORLs.*

Proof. Let $M_i = (N_i, \Sigma, \delta_i, S_i, F_i)$ be an OFSA that accepts language L_i for $i = 1, 2$. Define a new OFSA $M = (N, \Sigma, \delta, S, F)$ by setting $N = N_1 \sqcup N_2 \sqcup \{S\}$ (with any total order that preserves the order on N_1 and N_2),

$$\delta(A, a) = \begin{cases} \delta_1(A, a) & \text{if } A \in N_1 \\ \delta_2(A, a) & \text{if } A \in N_2 \\ \delta_1(S_1, a) \sqcup \delta_2(S_2, a) & \text{if } A = S \end{cases}$$

for all $A \in N$, $a \in \Sigma$ and

$$F = \begin{cases} F_1 \sqcup F_2 \sqcup \{S\} & \text{if } S_1 \in F_1 \text{ or } S_2 \in F_2 \\ F_1 \sqcup F_2 & \text{otherwise} \end{cases}.$$

It is easy to see that M accepts a word w if and only if $w \in L_1 \cap L_2$. Hence, $L_1 \cap L_2$ is an ORL. The negative results again follow from the previous lemma and De Morgan's law, together with the fact that every DORL is an ORL. \square

We conclude that determinism is a non-trivial restriction on ordinal regular languages.

Proposition 5.8. *The class of DORLs is a strict subclass of the class of ORLs.*

Proof. It is immediate by the definition that every DORL is an ORL. Conversely, the two classes cannot be the same, as only one of them is closed under complementation. \square

References

- [1] Hopcroft J. E., Ullman J. D., Motwani R. (2006) Introduction to Automata Theory, Languages, and Computation, Pearson/Addison-Wesley, 3rd ed.
- [2] Linz P. (2016) An Introduction to Formal Languages and Automata 6, Linz P.
- [3] Hamkins J. D., Lewis A. (2000) Infinite time Turing machines, Journal of Symbolic Logic 65: 567–604.
- [4] Carl M. (2019) Ordinal Computability. An Introduction to Infinitary Machines, De Gruyter
- [5] Löwe B. (2006) Space Bounds for Infinitary Computation, CiE
- [6] Carl M., Löwe B., Rin B.G. (2017) Koepke Machines and Satisfiability for Infinitary Propositional Languages, CiE
- [7] Perrin D., Pin J. (2004) Infinite words - Automata, Semigroups, Logic and Games, Pure and applied mathematics series