Bachelor thesis
Computing Science

Radboud University

Extending context-free grammars with conjunction and negation

Author:
Astrid van der Jagt
a.vanderjagt@student.ru.nl
s4571037

Supervisor:
prof. dr. Herman Geuvers
h.geuvers@cs.ru.nl

Assessor:
dr. Jurriaan Rot
jrot@cs.ru.nl

March 21, 2021

Abstract

Context-free grammars are used to inductively define languages with production rules. They are context-free in the sense that the production rules can be applied without regarding the context in which the application occurs. However, the current way of describing these grammars is limited, since they only use the operations constant languages, concatenation and disjunction. We show that conjunctive grammars extend the context-free grammars with the conjunction operation, while still maintaining the principle behind context-free grammars. Furthermore, we show that boolean grammars extend conjunctive grammars with the negation operation. We study which languages can be defined with these new grammar formalisms, and what it means exactly that a grammar defines a language.

Contents

Chapter 1

Introduction

Formal grammars are a formalism to describe languages mathematically. With *production rules* they define *properties* of what words or sentences are "allowed" in a language. Such a production rule is of the form $A \to p$, where p is a "pattern" of the words that can be produced by the auxiliary symbol A. The language defined by a grammar consists of the words that can be produced by the grammar. [19]

Context-free grammars describe context-free languages. The syntax of languages is defined inductively: properties of strings are logically determined by the properties of their substrings. The grammars are used to generate the strings over the input alphabet, denoted by Σ. The generation process transforms a string by applying a production rule. Suppose we have a production rule $A \to a$. We can apply this rule to the string uAv, where u is the *precontext* and v the *postcontext* of the symbol A. The application of the rule, denoted by the derivation sequence $uAv \Rightarrow uav$, produces the string uav.

These grammars are context-free in the sense that properties of substrings do not depend on the context in which they occur. In other words, a rule can always be applied, regardless of where the auxiliary symbol occurs. The context does not limit the applicability of the rule.

However, context-free grammars are limited; they only use the operations of constant languages, concatenation, and disjunction (represented by the fact that it is possible to define multiple rules for one auxiliary symbol). When there are multiple rules for an auxiliary symbol A, only one of the rules can be applied. There is no way of generating words that are accepted in a language only if they satisfy multiple properties at the same time.

Similarly, context-free grammars also do not provide a way to generate words that are accepted in a language only if they satisfy a property α_1, and explicitly do not satisfy another property α_2.

Scheinberg [16] has shown that the conjunction and negation operations cannot be represented through disjunction, since the intersection of two context-free languages or the complement of a context-free language is not necessarily context-free.

Okhotin [12] has introduced the concept of conjunctive grammars in 2001. These are essentially context-free grammars extended with the conjunction operation. He argues that these grammars still preserve the general intention of context-free grammars: the deduction of the properties of a string does not depend on the context in which it occurs. The conjunctive grammars offer the possibility of defining words that need to satisfy multiple properties.

Furthermore, in 2004 Okhotin [10] has defined the concept of boolean grammars, an extension of conjunctive grammars. Boolean grammars include the negation operation, providing a method to define words that should satisfy some property, and explicitly not satisfy some other property.

These extensions of context-free grammars allow to define more languages that, according to the Chomsky hierarchy, are context-sensitive languages [2]. Context-sensitive grammars have proven to be hard to parse [3], and are therefore hardly ever used. The membership problem (whether a string can be generated by a grammar) for context-sensitive grammars is in the computational complexity class **PSPACE** [6].

The membership problem for conjunctive and boolean grammars is in the computational complexity class **P** [10]. These extensions can be seen as a grammar type between context-free and context-sensitive grammars, as they are less hard to parse than context-sensitive grammars, when we assume that $\mathbf{P} \neq \mathbf{PSPACE}$. That is why Okhotin proposes a different hierarchy of formal grammars, based on their computational complexity classes. This is further described in Section 7.2.

This thesis explores these two extensions of context-free grammars as defined by Okhotin. We provide some in-depth examples to illustrate the possibilities and limitations.

In Part I, some mathematical background is given. In Chapter 2 we provide relevant order theory. In Chapter 3 we discuss formal languages and operations over those languages. Finally, in Chapter 4 we give three equivalent definitions of how the meaning of a context-free grammar can be described.

In Part II the extensions are discussed. Chapter 5 elaborates on the conjunctive grammars, defined by the same three methods as context-free grammars. Furthermore, we prove for one conjunctive grammar what language exactly it produces. Chapter 6 discusses the boolean grammars. We show that constructing a sound definition of boolean grammars is far from trivial.

In Chapter 7, related research is evaluated: an alternative method to define boolean grammars based on three-valued languages by Kountouriotis [7] and a new proposed hierarchy for formal grammars given by Okhotin [10].

The conclusions of this thesis can be found in Chapter 8.

Part I

Preliminaries

Chapter 2

Order Theory

Order theory can be used as a mathematical foundation for understanding the meaning of context-free grammars. Because of the properties of partially ordered sets and complete lattices, we are able to study precisely how context-free grammars define languages inductively.

Ordering elements of a set occurs frequently. We order the letters of the alphabet, to help us find a word in the dictionary efficiently. Seat numbers in a concert hall are ordered, so that attendees of a concert know where there seats are positioned.

The ordering of a set is based on a relation on the elements of the set. This relation defines for every pair of elements what their position is with respect to each other. An elements can precede another element, based on how the relation is defined.

Definition 2.1 (*Partial ordering [15]*)

A relation R on a set S is called a *partial ordering* if it is:

- reflexive: $R(a, a)$, for all $a \in S$
- antisymmetric: if $R(a, b) \wedge R(b, a)$ then $a = b$, for all $a, b \in S$
- transitive: if $R(a, b) \wedge R(b, c)$ then $R(a, c)$, for all $a, b, c \in S$

Definition 2.2 (*Partially ordered set/poset [15]*)

A *partially ordered set* or *poset* is a set (S, R), where S is a set and R is a partial ordering on that set. Members of S are called elements of the poset. We write $a \sqsubseteq b$ to denote that $R(a, b)$ in an arbitrary poset (S, R), with R as some ordering $\sqsubseteq$. We say "a is less or equal than b" or "b is greater or equal than a" if $a \sqsubseteq b$.

A commonly used example of a partial ordering is the relation $\leq$ on the set of integers ($\mathbb{Z}$). The relation is reflexive: for all integers $a \leq a$. The relation is antisymmetric: when $a \leq b$ and $b \leq a$, we know that $a = b$. Finally, it is a transitive relation. If $a \leq b$ and $b \leq c$, then we know that $a \leq c$. It follows that $(\mathbb{Z}, \leq)$ is a poset.

Definition 2.3 (*Upper bound/lower bound [15]*)

An element u of a poset $(S, \sqsubseteq)$ such that $a \sqsubseteq u$ for all elements $a \in A$, where A is a subset of S, is called an *upper bound* of A. Likewise, an element l of a poset $(S, \sqsubseteq)$ such that $l \sqsubseteq a$ for all elements $a \in A$, where A is a subset of S is called a *lower bound* of A.

Definition 2.4 (*Least upper bound/greatest lower bound [15]*)

The element x is called the *least upper bound* of the subset A of S, denoted by $\bigvee A$, if

- x, is an upper bound of A and
- $x \sqsubseteq z$, for all z, where z is an upper bound of A.

Similarly, the element y is called the *greatest lower bound* of A, denoted by $\bigwedge A$, if

- y is a lower bound A and
- $z \sqsubseteq y$, for all z, where z is a lower bound of A.

Definition 2.5 (*Chain [8]*)

A subset A of S of a poset $(S, \sqsubseteq)$ is called a *chain* if it is consistent in the sense that if we take any two elements of $a, b \in A$, we have $a \sqsubseteq b$ or $b \sqsubseteq a$.

Definition 2.6 (*Lattice [15]*)

A partially ordered set $(S, \sqsubseteq)$ in which every pair of elements has both a least upper bound and a greatest lower bound is called a *lattice*. A lattice is called *complete* if all the subsets X of S have both a least upper bound and a greatest lower bound, denoted by $\bigvee X$ and $\bigwedge X$.

Definition 2.7 (*Bottom [8]*)

The *bottom* $\bot$ is a least element of a partially ordered set $(S, \sqsubseteq)$, such that $\bot \sqsubseteq a$ for every element $a \in S$. The bottom element is unique if it exists. In a Hasse diagram this is the lowest element.

Proposition 2.8 (*Every complete lattice has a bottom [8]*)

Every complete lattice $(S, \sqsubseteq)$ has a bottom element given by $\bot = \bigvee \emptyset$.

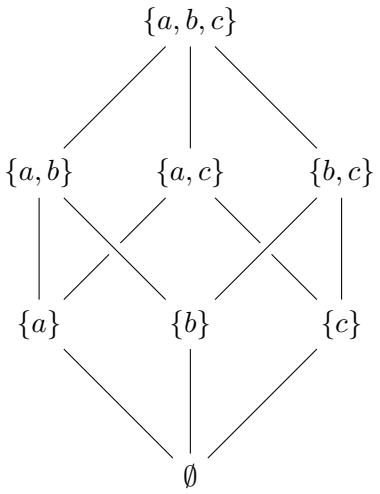
We can often represent a poset with a *Hasse diagram*. This is a directed graph where the nodes are the element of the set, and the edges represent the relation on the elements. The reflexive relations and transitive relations are ommitted from the graph. Finally, the direction of the edges is upward; for every relation $a \prec b$, element a is positioned below element b, and there is an edge between the two nodes.

Proposition 2.9 (*Poset $(\mathcal{P}(S), \subseteq)$ is a complete lattice [15]*)

The poset $(\mathcal{P}(S), \subseteq)$ is a complete lattice for every set S: the least upper bound of $A \subseteq S$ and $B \subseteq S$ is $A \cup B$, and the greatest lower bound is $A \cap B$.

Example 2.10 (*Hasse diagram*)

We consider the poset $(\mathcal{P}\{a, b, c\}, \subseteq)$. The Hasse diagram of this poset is:



A Hasse diagram can be used to determine the upper and lower bounds of a set. The upper bounds of the subset $\{\emptyset, \{a\}, \{c\}\}$ in Example 2.10 are $\{a, c\}$ and $\{a, b, c\}$. We follow the edges upward and only include the edges that can be reached by every node of the subset. The only lower bound of this subset is $\emptyset$.

In Section 4.3 we define the context-free grammars in terms of language equations. In order to solve the system of equations we consider a function that needs to be monotonically increasing and Scott continuous, to find the least fixed point of that function.

Definition 2.11 (*Monotonically increasing function [8]*)

We consider the poset $(S, \sqsubseteq)$. A function $F : S \to S$ is said to be *monotonically increasing* if for all x, y it is the case that $x \sqsubseteq y$ implies $F(x) \sqsubseteq F(y)$.

Definition 2.12 (*Scott Continuity [17]*)

We consider the poset $(S, \sqsubseteq)$. A function $F : S \to S$ is *Scott continuous* if it is monotonically increasing and if for every chain $x_1 \sqsubseteq x_2 \sqsubseteq ...$ we have $F(\bigvee_i x_i) = \bigvee_i F(x_i)$.

Definition 2.13 (*Fixed point [8]*)

Let F be a function. A *fixed point* of F is an element x, for which $F(x) = x$. A least fixed point of F is a fixed point x, such that for any fixed point y of F, it is the case that $x \sqsubseteq y$.

Theorem 2.14 (*Kleene's Fixed Point Theorem*)

Let $(S, \sqsubseteq)$ be a partially ordered set that is a complete lattice. Let $F : S \to S$ be a Scott continuous function. Then F has a least fixed point in S: there is a $X \in S$, such that $F(X) = X$, and such that if $F(Y) = Y$, then $X \sqsubseteq Y$.

Proof:
We choose $X = \bigvee_{i \in \mathbb{N}} F^i(\bot)$.

$$
\begin{aligned}
F(X) &= F\left(\bigvee_{i \in \mathbb{N}} F^i(\bot)\right) && \text{by substituting } X \\
&= \bigvee_{i \in \mathbb{N}} F^{i+1}(\bot) && \text{because } F \text{ is Scott continuous} \\
&= \bigvee_{i \in \mathbb{Z}^+} F^i(\bot) && \text{since } (\mathbb{N} \setminus 0) \text{ is } \mathbb{Z}^+ \\
&= \{\bot\} \cup \bigvee_{i \in \mathbb{Z}^+} F^i(\bot) && \text{by definition of } \bot \\
&= F^0(\bot) \cup \bigvee_{i \in \mathbb{Z}^+} F^i(\bot) && \text{because } F^0(\bot) = \bot \\
&= \bigvee_{i \in \mathbb{N}} F^i(\bot) && \text{since } (\mathbb{Z}^+ \cup 0) \text{ is } \mathbb{N} \\
&= X
\end{aligned}
$$

Since it is the case that $F(X) = X$, we have a fixed point. The question remains whether it is a least fixed point.

Now suppose we have a different fixed point Y, and thus $F(Y) = Y$. By

induction we can prove that $F^i(\bot) \sqsubseteq Y$, for all $i \in \mathbb{N}$.

Base case: $i = 0$.
We get $F^0(\bot) = \bot$, and $\bot \sqsubseteq Y$, so this holds.

Induction step: $i = i + 1$.
Suppose that for some $i \in \mathbb{N}$ it holds that $F^i(\bot) \sqsubseteq Y$ (Induction hypothesis).

We need to show that $F^{i+1}(\bot) \sqsubseteq F(Y) = Y$.

Since F is monotone, and because the Induction hypothesis states that $F^i(\bot) \sqsubseteq Y$, we can write $F(F^i(\bot)) \sqsubseteq F(Y)$. Since Y is a fixed point, we have $F^{i+1}(\bot) = F(F^i(\bot) \sqsubseteq F(Y) = Y$.

This concludes the induction step.

Now that we have proven that $F^i(\bot) \sqsubseteq Y$, for all $i \in \mathbb{N}$, it must be the case that $\bigsqcup_{i \in \mathbb{N}} F^i(\bot) \sqsubseteq Y$. This means that $X \sqsubseteq Y$. We can conclude that X is therefore the least fixed point of F.

This proves Theorem 2.14.

Chapter 3

Languages

In this chapter, we define languages mathematically, based on set theory. We introduce notation that is used throughout this thesis.

Definition 3.1 (*Alphabet [19]*)

The *alphabet* of a language consists of the symbols that are used by the language, it is denoted by Σ. A *string over an alphabet* is a finite sequence of symbols $\in \Sigma$.

We use the notation $|w|$ for a string w to denote the length of the string, the number of symbols in w. We write $|w|_a$ to denote the number of a's used in word w.

Definition 3.2 (*Empty string [19]*)

The word of length 0 is the *empty string*. We denote the empty string with the symbol ε.

Definition 3.3 (*Set of strings over Σ [19]*)

Let Σ be an alphabet. The set Σ^* is defined recursively:

- Basis: $\varepsilon \in \Sigma^*$.
- Recursive step: if $a \in \Sigma$ and $w \in \Sigma^*$, then $aw \in \Sigma^*$.

If Σ has n elements, Σ^* has n^ℓ elements of length ℓ.

Definition 3.4 (*Strings of length up to ℓ [19]*)

Let Σ be an alphabet. We define $\Sigma^{\leq \ell}$ as the set of strings over Σ of length up to and including ℓ.

With these definitions, we are now able to define the concept of languages in terms of set theory:

Definition 3.5 (*Language [19]*)

A *language* $\mathrm{L} \subseteq \Sigma^*$ is a set of strings, consisting of symbols from the alphabet of the language Σ.

We use the set-builder notation to describe languages: $\{variable \mid predicate\}$. All the variables that hold according to the predicate are members of the set.

There are several operations we can perform over words and languages. Below follow definitions of these operations.

Definition 3.6 (*Union of languages [19]*)

Let X and Y be languages. The *union* of these two languages $\mathrm{X} \cup \mathrm{Y}$ is the language $\{w \mid w \in \mathrm{X} \vee w \in \mathrm{Y}\}$.

Definition 3.7 (*Intersection of languages [19]*)

Let X and Y be languages. The *intersection* of these two languages $\mathrm{X} \cup \mathrm{Y}$ is the language $\{w \mid w \in \mathrm{X} \wedge w \in \mathrm{Y}\}$.

Definition 3.8 (*Complement of a language [10]*)

Let L be a language with alphabet Σ. We define $\overline{\mathrm{L}}$ as the *complement* of the language: for every word w in Σ^*, $w \notin \mathrm{L}$ implies $w \in \overline{\mathrm{L}}$.

Definition 3.9 (*Concatenation of words and languages [19]*)

Let $x, y \in \Sigma^*$. The *concatenation* of x and y is written $w = xy$. The resulting word consists of the symbols of x followed by the symbols of y.

Let X and Y be languages. The concatenation of these two languages XY is the language $\{w = xy \mid x \in \mathrm{X} \wedge y \in \mathrm{Y}\}$.

Concatenating X n times with itself is denoted by X^n, and $\mathrm{X}^0 = \{\varepsilon\}$.

Note that when $\mathrm{Y} = \emptyset$, concatenating with another language X will give $\mathrm{XY} = \emptyset$, since there is no $y \in \mathrm{Y}$ such that $w = xy$.

Definition 3.10 (*Kleene's star [19]*)

Let X be a set. Then

$$\mathrm{X}^* = \bigcup_{i=0}^{\infty} \mathrm{X}^i \qquad \textit{where } ^* \textit{ is Kleene's star}$$

is the set of all strings that can be built from the elements of X.

Definition 3.11 (*Reversal of a word [19]*)

Let $w \in \Sigma^*$. The *reversal* of w is written as w^R: the symbols of w are reversed: $(w_1...w_n)^R = w_n...w_1$.

In this thesis, we avoid using parentheses when not needed. We use the following precedence of operations, starting from the highest precedence:

1. Concatenation
2. Negation, denoted by $\neg$
3. Conjunction, denoted by & or $\wedge$
4. Disjunction, denoted by $\vee$

This means that the expression $\neg$XY & $\neg$X $\vee$ Y has the same meaning as $((\neg(\mathrm{XY}))\ \&\ (\neg\mathrm{X})) \vee \mathrm{Y}$.

Example 3.12

Let $\mathrm{X} = \{a, b, c\}$ and $\mathrm{Y} = \{a, d, e, f\}$. Then

$$
\begin{aligned}
\mathrm{X} \cup \mathrm{Y} &= \{a, b, c, d, e, f\} \\
\mathrm{X} \cap \mathrm{Y} &= \{a\} \\
\mathrm{XY} &= \{aa, ad, ae, af, ba, bd, be, bf, ca, cd, ce, cf\} \\
\mathrm{X}^0 &= \{\varepsilon\} \\
\mathrm{X}^1 &= \{a, b, c\} \\
\mathrm{X}^2 &= \{aa, ab, ac, ba, bb, bc, ca, cb, cc\} \\
\mathrm{X}^{\leq 2} &= \{a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc\} \\
\mathrm{X}^* &= \mathrm{X}^0 \cup \mathrm{X}^1 \cup ... = \\
&\quad \{a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, ...\}
\end{aligned}
$$

Chapter 4

Context-free grammars

Context-free grammars (CFGs) are used to describe context-free languages. The syntax of languages is defined inductively, properties of strings are logically determined by the properties of their substrings [19]. These grammars are context-free in the sense that properties of substrings do not depend on the context in which they occur. In other words, a rule can always be applied, regardless of the situation.

An example of a context-free language is the language that contains all the palindromes over the input alphabet $\{a, b\}$. The words $abba$ and $babab$ are in the language, while the word $aabb$ is not in the language. The context-free grammar that produces this language starts with a start symbol S and generates the words inductively. Symbol S can produce one of the terms $a\mathrm{S}a, b\mathrm{S}b, a, b$ or the empty word. The term $a\mathrm{S}a$ produces one of the terms $aa\mathrm{S}aa, ab\mathrm{S}ba, aaa, aba$ or aa. All the terms without the symbol S are words that are in the language generated by the grammar.

We will call this context-free language L_1. It is used as an example throughout this chapter to illustrate three equivalent methods [11] to define what language a context-free grammar generates. Defining the meaning of a grammar in terms of rewriting with production rules is described in Section 4.1. The second method, in terms of a formal deduction system with inference rules can be found in Section 4.2. Finally, Section 4.3 defines the meaning of a grammar in terms of language equations.

4.1 Context-free grammars in terms of Rewriting

The rewriting method originates from Chomsky [2]. A production rule $A \to \alpha$ states that the nonterminal A can be rewritten to the term α. Starting from the startsymbol, nonterminals can be rewritten according to the production rules, until the resulting term consists solely of terminal characters: a word that is generated by the grammar.

Definition 4.1 (*Context-free grammars in terms of Rewriting [19]*)

A *context-free grammar* is a quadruple $G = (\Sigma, N, R, S)$, where

- Σ is the alphabet: a finite set of symbols, also called *terminal* characters, that are used in the language described by the grammar. The words in the described language are only allowed to consist of these symbols.
- N is a finite set of *nonterminal* symbols, which are used in the grammar as 'building stones' of a word. These symbols do not occur in accepted words of a language.
- R is a finite set of *production rules*, each of the form

$$A \ \to \ \alpha$$

  where $A \in N$, and $\alpha \in (\Sigma \cup N)^*$.
- $S \in N$ is the *startsymbol*, a nonterminal that represents the set of words that are accepted in the defined language.

A language generated by a context-free grammar is called a *context-free language*. A word $w \in \Sigma^*$ is in the language generated by the grammar when a derivation sequence (Definition 4.2) $S \Rightarrow^* w$ can be constructed (where S is the startsymbol of the grammar).

When there are two rules in a grammar over the same nonterminal, e.g. $A \to \alpha_1$ and $A \to \alpha_2$, it is denoted as $A \to \alpha_1 \mid \alpha_2$. The $\mid$ symbol denotes disjunction. In a word uAv, where u is the *precontext* and v the *postcontext*, we can apply only one of the rules. This is regardless of what u and v are, which is why it is called a context-free grammar.

A production rule of the form $A \to uAv$ is called a *recursive rule*, since it defines the nonterminal A in terms of itself.

Definition 4.2 (*Derivation sequence [19]*)

We define how words are generated by the context-free grammar with a *derivation sequence*. We use the notation $w \Rightarrow v$ for each step, if $w = x\mathrm{A}y$ and $v = x\alpha y$ for some $x, y, \alpha \in (\Sigma \cup N)^*$, $\mathrm{A} \in N$ and production rule $\mathrm{A} \to \alpha$ in R.

We use $\Rightarrow^n$ to denote that there are a exactly n steps between the left-side of the arrow and the right-side of the arrow.

We use $\Rightarrow^*$ to denote that there are a finite amount of steps between the left-side of the arrow and the right-side of the arrow.

Example 4.3

The following context-free grammar G_1 generates the language $\mathrm{L}_1 = \{w \mid w = w^R \wedge w \in \{a, b\}^*\}$. This language contains palindromes with the input alphabet $\Sigma = \{a, b\}$.

$$\mathrm{S} \;\to\; a\mathrm{S}a \mid b\mathrm{S}b \mid a \mid b \mid \varepsilon$$

We can construct the following derivation sequence to show that the word *abba* is accepted in the language, according to the grammar:

$$\mathrm{S} \Rightarrow a\mathrm{S}a \Rightarrow ab\mathrm{S}ba \Rightarrow abba$$

4.2 Context-free grammars in terms of a Formal Deduction System

We can describe the meaning of a context-free grammar in terms of a formal deduction system. Showing how words are generated from the grammar is done with an inference tree. The tree is built by applying the rules, and it is finished if all the branches of the tree end in axioms.

Definition 4.4 (*Formal deduction system for context-free grammars [18]*)

For every context-free grammar $G = (\Sigma, N, R, S)$, a formal deduction system $\vdash_G$ can be defined as follows. Every production rule of the form $\mathrm{A} \to \alpha_1 ... \alpha_n$ has a corresponding inference rule:

$$\frac{\alpha_1(u_1) \quad ... \quad \alpha_n(u_m)}{\mathrm{A}(w)} \; [\textit{rulename1}]$$

Where $w = u_1 ... u_m$ is in Σ^*. Note that the different u_i don't overlap. The term w is "sliced" and every α_j is applied over a part of w. The statement below the line is the *conclusion* of the rule, it holds when all the statements above the line, the *premises* hold as well. The rule is provided with a rule name in brackets, to add clarity.

Axioms of the grammar are written for every $a \in \Sigma$ and for rules that have a ε on the right-hand side of the arrow:

$$\vdash a(a) \qquad [\textit{rulename2}]$$
$$\vdash \mathrm{A}(\varepsilon) \qquad [\textit{rulename3}]$$

Since axioms do not depend on other statements in order to be true, we denote $\vdash$ to show that the statement holds.

A word $w \in \Sigma^*$ is in the language generated by the grammar when an inference tree (Definition 4.5) can be constructed with $S(w)$ as its conclusion (where S is the startsymbol of the grammar).

Note that the terms above the line in an inference rule are never larger than the one below the line. So in [*rulename1*] it must be the case that for every i, $|u_i| \leq |w|$.

Definition 4.5 (*Inference tree [18]*)

We define how words are generated by the grammar with an *inference tree*. The bottom or *root* of the tree states the conclusion. Inference rules are applied on statements. You can prove these statements — show that a word can be produced by a grammar — by constructing an inference tree, where every statement at the top of the tree is an axiom.

When we read the tree bottom up, we start at the root of the tree and follow every "branch" of the tree, until it ends in an axiom.

Example 4.6

Recall the context-free language $\mathrm{L}_1 = \{w \mid w = w^R \wedge w \in \{a, b\}^*\}$, that contains palindromes with the input alphabet $\Sigma = \{a, b\}$. We define the corresponding formal deduction system $\vdash_{G_1}$, with the inference rules given in Table 4.1, where $w \in \Sigma^*$.

The word *abba* is accepted in the language, as we show with the following inference tree: (the rule names of the axioms are ommitted)

$$\dfrac{a(a) \qquad \dfrac{b(b) \quad \mathrm{S}(\varepsilon) \quad b(b)}{\mathrm{S}(bb)}\,[\mathrm{S}^4] \qquad a(a)}{\mathrm{S}(abba)}\,[\mathrm{S}^3]$$

Rule	Rule name
$\vdash a(a)$	$[\mathrm{axiom}^a]$
$\vdash b(b)$	$[\mathrm{axiom}^b]$
$\vdash \mathrm{S}(\varepsilon)$	$[\mathrm{S}^0]$
$\dfrac{a(a)}{\mathrm{S}(a)}$	$[\mathrm{S}^1]$
$\dfrac{b(b)}{\mathrm{S}(b)}$	$[\mathrm{S}^2]$
$\dfrac{a(a) \quad \mathrm{S}(w) \quad a(a)}{\mathrm{S}(awa)}$	$[\mathrm{S}^3]$
$\dfrac{b(b) \quad \mathrm{S}(w) \quad b(b)}{\mathrm{S}(bwb)}$	$[\mathrm{S}^4]$

Table 4.1: The inference rules for the context-free grammar $\vdash_{G_1}$

An important property of context-free grammars in terms of a deduction system is that when an inference rule results in multiple branches (when reading the rule bottom up), the terms do not overlap. For example, rule

[S^3], when applied over some word awa, the word is sliced into three terms.

The grammar from Example 4.6 is quite simple, because there is no ambiguity in how words should be sliced, when reading the rules bottom up, because of the sharp definition. However, there are context-free grammars for which it is not so clear how to slice the words. Below we illustrate such an example, in Example 4.7.

Example 4.7

The context-free grammar G_2 generates the language
$L_2 = \{xyc \mid x \in \{a,b\}^* \wedge y \in \{a,b,c\}^* \wedge |x| = |y|_c + 1 \wedge |x| \geq 1 \wedge |y| \geq 0\}$
with the input alphabet $\Sigma = \{a,b,c\}$:

$$\begin{aligned} S &\rightarrow aSX \mid bSX \mid ac \mid bc \\ X &\rightarrow aX \mid bX \mid c \end{aligned}$$

Table 4.2 shows the inference rules for the corresponding formal deduction system $\vdash_{G_2}$, with $w \in \Sigma^*$.

Rule		**Rule name**
$\vdash a(a)$		[axiom^a]
$\vdash b(b)$		[axiom^b]
$\vdash c(c)$		[axiom^c]
$\dfrac{a(a) \quad S(u_1) \quad X(u_2)}{S(aw)}$	where $w = u_1u_2$	[S^1]
$\dfrac{b(b) \quad S(u_1) \quad X(u_2)}{S(bw)}$	where $w = u_1u_2$	[S^2]
$\dfrac{a(a) \quad c(c)}{S(ac)}$		[S^3]
$\dfrac{b(b) \quad c(c)}{S(bc)}$		[S^4]
$\dfrac{a(a) \quad X(w)}{X(aw)}$		[X^1]
$\dfrac{b(b) \quad X(w)}{X(bw)}$		[X^2]
$\dfrac{c(c)}{X(c)}$		[X^3]

Table 4.2: The inference rules for the context-free grammar $\vdash_{G_2}$

Definition 4.8 (*Parsing a word for a CFG*)

Parsing a word $w \in \Sigma^*$ for a context-free grammar is deciding whether there exists an inference tree with the term $S(w)$ as its conclusion, where S is the startsymbol of the grammar, and also providing the specific inference tree.

Suppose we want to construct an inference tree to show that the word $baacacbc \in \mathrm{L}_2$. Only the $[\mathrm{S}^2]$ rule can be applied, because we need to start with the startsymbol and this is the only S rule that fits. The $[\mathrm{S}^2]$ rule requires us to slice the word in three parts: the first part is easy, we take the b.

That leaves us with the remainder of the word: $aacacbc$. The rule does not specify where we should split the word. There are eight ways to cut the word in two slices u_1, (that requires an S rule) and u_2 (that requires an X rule):

1. $u_1 = \varepsilon, u_2 = aacacbc$
2. $u_1 = a, u_2 = acacbc$
3. $u_1 = aa, u_2 = cacbc$
4. $u_1 = aac, u_2 = acbc$
5. $u_1 = aaca, u_2 = cbc$
6. $u_1 = aacac, u_2 = bc$
7. $u_1 = aacacb, u_2 = c$
8. $u_1 = aacacbc, u_2 = \varepsilon$

Observing that there is no axiom for ε, we can exclude option 1 and option 8. That still leaves us with six ways to parse $aacacbc$. We can conclude that parsing context-free grammars can be challenging, due to the fact that it is not always clear how to slice the words.

4.3 Context-free grammars in terms of Language Equations

The meaning of a context-free grammar can be defined in terms of a system of equations with languages as unknowns.

Definition 4.9 (*Constant languages [19]*)

Let Σ be an alphabet. We write $\{\alpha\}$ for each $\alpha \in \Sigma$ and $\{\varepsilon\}$ to denote the *constant languages*. These languages are not recursively defined.

Definition 4.10 (*Language equations for context-free grammars [1, 4]*)

For every context-free grammar $G = (\Sigma, N, R, S)$, a system of equations with languages as unknowns $\mathcal{E}_G$ can be defined as follows. We write

$$\mathrm{A} = \bigcup_{\mathrm{A} \to X_1 \ldots X_l \in R} X_1 \cdot \ldots \cdot X_l$$

for all nonterminals $\mathrm{A} \in N$ as unknown languages, and for all $a \in \Sigma \cup \{\varepsilon\}, a$ denotes $\{a\}$.

Such a system has a least solution, which is the least fixed point of the function derived from the grammar (see Corollary 4.15). A word $w \in \Sigma^*$ is in the language generated by the grammar when it is in the least solution (of the startsymbol).

Example 4.11

Recall the context-free language $\mathrm{L}_1 = \{w \mid w = w^R \wedge w \in \{a, b\}^*\}$, that contains palindromes with the input alphabet $\Sigma = \{a, b\}$. We define the corresponding system of language equations $\mathcal{E}_{G_1}$:

$$\mathrm{S} = \{a\}\mathrm{S}\{a\} \cup \{b\}\mathrm{S}\{b\} \cup \{a\} \cup \{b\} \cup \{\varepsilon\}$$

We consider the function $F : \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Sigma^*)$ for Example 4.11 derived from the grammar:

$$F(\mathrm{S}) = \{a\}\mathrm{S}\{a\} \cup \{b\}\mathrm{S}\{b\} \cup \{a\} \cup \{b\} \cup \{\varepsilon\}$$

We want to find the least solution of F, a least fixed point. We can do this by iterating with the empty set as start. We get:

- $F^0(\emptyset) = \emptyset$
- $F^1(\emptyset) = \{a\}\emptyset\{a\} \cup \{b\}\emptyset\{b\} \cup \{a\} \cup \{b\} \cup \{\varepsilon\} = \{a, b, \varepsilon\}$
- $F^2(\emptyset) = F(\{a, b, \varepsilon\}) = \{a\}\{a, b, \varepsilon\}\{a\} \cup \{b\}\{a, b, \varepsilon\}\{b\} \cup \{a\} \cup \{b\} \cup \{\varepsilon\}$
  $= \{a, b, \varepsilon, aaa, aba, aa, bab, bbb, bb\}$

- ...

We observe that $F^i(\emptyset)$ contains the words that can be generated by applying the production rules of the grammar at most i times.

Lemma 4.12 (*CFG - Monotonically increasing function*)

Let $F : \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Sigma^*)$ be the function with $F(\mathrm{S}) = \{a\}\mathrm{S}\{a\} \cup \{b\}\mathrm{S}\{b\} \cup \{a\} \cup \{b\} \cup \{\varepsilon\}$ and $\Sigma = \{a, b\}$. Function F is a monotonically increasing function.

Proof:
Let $w \in F(\mathcal{A})$, and $\mathcal{A} \subseteq \mathcal{B}$. There are three cases to be considered:

1. $w \in \{a, b, \varepsilon\}$. Then $w \in F(\mathcal{B})$.
2. w is of the form ava, with $v \in \mathcal{A}$. We know that v is also in $\mathcal{B}$, because $\mathcal{A} \subseteq \mathcal{B}$. Then $ava \in F(\mathcal{B})$.
3. w is of the form bvb, with $v \in \mathcal{A}$. We know that v is also in $\mathcal{B}$, because $\mathcal{A} \subseteq \mathcal{B}$. Then $bvb \in F(\mathcal{B})$.

This proves Lemma 4.12.

With every iteration it is the case that $F^i(\emptyset) \subseteq F^{i+1}(\emptyset)$, because the function is monotonically increasing with respect to the partial ordering $(\mathcal{P}(\Sigma^*), \subseteq)$. In other words, we observe that $F^0(\emptyset) \subseteq F^1(\emptyset) \subseteq F^2(\emptyset) \subseteq ...$

Example 4.13

Let $F : \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Sigma^*)$ be the function with $\Sigma = \{a, b\}$ and $F(\mathrm{S}) = \{a\}\mathrm{S}\{a\} \cup \{b\}\mathrm{S}\{b\} \cup \{a\} \cup \{b\} \cup \{\varepsilon\}$.

Scott continuity will be used for a specific chain, namely $F^0(\emptyset) \subseteq F^1(\emptyset) \subseteq F^2(\emptyset) \subseteq ... = \emptyset \subseteq \{a, b, \varepsilon\} \subseteq \{a, b, \varepsilon, aaa, aba, aa, bab, bbb, bb\} \subseteq ...$
For this particular chain we will verify in detail that $F(\bigcup_i F^i(\emptyset)) = \bigcup_i F(F^i(\emptyset))$.

$$\begin{aligned} F(\bigcup_i F^i(\emptyset)) &= F(\emptyset \cup \{a, b, \varepsilon\} \cup \{a, b, \varepsilon, aaa, aba, aa, bab, bbb, bb\} \cup ...) \\ &= F(\{w \mid w = w^R \wedge w \in \{a, b\}^*\}) \\ &= \{a\}\{w \mid w = w^R\}\{a\} \cup \{b\}\{w \mid w = w^R\}\{b\} \cup \{a\} \cup \{b\} \cup \{\varepsilon\} \\ &= \{w \mid w = w^R\} \end{aligned}$$

$$\begin{aligned} \bigcup_i F(F^i(\emptyset)) &= F(\emptyset) \cup F(\{a, b, \varepsilon\}) \cup F(\{a, b, \varepsilon, aaa, aba, aa, bab, bbb, bb\}) \cup ... \\ &= \{a, b, \varepsilon\} \cup \{a, b, \varepsilon, aaa, aba, aa, bab, bbb, bb\} \cup ... \\ &= \{w \mid w = w^R\} \end{aligned}$$

Proposition 4.14 (*Scott continuity in operations of CFGs [17]*)

For a partially ordered set $(\mathcal{P}(\Sigma^*), \subseteq)$ that is a complete lattice, if $F : \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Sigma^*)$ is a function defined from constant languages, concatenation, and disjoint union, then F is a Scott continuous function.

Because of Proposition 4.14 we can apply Kleene's Fixed Point Theorem to the system of language equations for context-free grammars, to find the solution of the system.

Corollary 4.15 (*CFG - Least solution*)

Since $(\mathcal{P}(\Sigma^*), \subseteq)$ is a partially ordered set and a complete lattice, we know for every context-free grammar $\mathcal{E}_G$ that the *least solution* of the language equations is $\bigcup_{k\geq 0} F^k(\emptyset)$, with the Scott continuous function $F : \mathcal{P}(\Sigma^*)^{|N|} \to \mathcal{P}(\Sigma^*)^{|N|}$ over the nonterminals N of G, derived from the system of equations $\mathcal{E}_G$. The least solution consists of exactly the languages generated by the nonterminals.

Example 4.16

The solution for the grammar $\mathcal{E}_{G_1}$ is $\bigcup_{i\geq 0} F^i(\emptyset) = \{w \mid w = w^R \wedge w \in \{a, b\}^*\}$:

$$\begin{aligned} F(\{w \mid w = w^R\}) &= \{a\}\{w \mid w = w^R\}\{a\} \cup \{b\}\{w \mid w = w^R\}\{b\} \cup \\ &\quad \{a\} \cup \{b\} \cup \{\varepsilon\} \\ &= \{w \mid w = w^R\} \end{aligned}$$

because $\{a\}\{w \mid w = w^R\}\{a\} \cup \{b\}\{w \mid w = w^R\}\{b\}$ is in $\{w \mid w = w^R\}$, and $\{a\}, \{b\}, \{\varepsilon\}$ are also in $\{w \mid w = w^R\}$.

In conclusion, $\{w \mid w = w^R\}$ is a fixed point of F and it can be shown that it is the least fixed point of F, and therefore the least solution.

Part II

Extensions of context-free grammars

Chapter 5

Conjunctive Grammars

Conjunctive grammars (CGs) were introduced by Okhotin in 2001 [12]. He defines conjunctive grammars as context-free grammars extended with the conjunction operation. This extension provides a method to generate words that have to satisfy multiple conditions in order to be in the language described.

The simplest application of conjunctive grammars is to define a grammar that generates the intersection of two context-free grammars. For example, the intersection of the context-free languages
$\{a^i b^j c^k \mid j = k \wedge i, j, k \geq 0\}$ and $\{a^i b^j c^k \mid i = j \wedge i, j, k \geq 0\}$ results in the language $\{a^n b^n c^n \mid n \geq 0\}$. This language cannot be generated by a context-free grammar [20]. However, conjunctive grammars also offer the possibility to use conjunction more freely, like disjunction is used in context-free grammars.

Similarly to context-free grammars, we will describe the meaning of conjunctive grammars using three equivalent definitions. Section 5.1 defines the meaning of conjunctive grammars in terms of rewriting. This is the most intuitive definition. We provide a proof to show for one conjunctive grammar what language it produces. In Section 5.2 we discuss the meaning of conjunctive grammars in terms of a formal deduction system. Finally, Section 5.3 describes the most important meaning, it is in terms of language equations.

All the definitions in this chapter are retrieved from Okhotin's paper [11]. With examples we will illustrate some of the possibilities that conjunctive grammars offer.

5.1 Conjunctive grammars in terms of Rewriting

The definition is almost the same as the definition of context-free grammars (Definition 4.1). The big difference is that the conjunction operation is allowed in conjunctive grammars, where it is not in context-free grammars. This allows to define production rules where multiple conditions have to be met.

Definition 5.1 (*Conjunctive Grammars in terms of Rewriting [11]*)

A *conjunctive grammar* is a quadruple $G = (\Sigma, N, R, S)$ where:

- Σ is the alphabet.
- N is a finite set of nonterminal symbols.
- R is a finite set of production rules of the form

$$\mathrm{A} \rightarrow \alpha_1 \,\&\, \alpha_2 \,\&...\&\, \alpha_n$$

  where $\mathrm{A} \in N,\ n \geq 1$ and $\alpha_k \in (\Sigma \cup N)^*$, with $1 \leq k \leq n$. Every α_i is called a *conjunct*. The & symbol denotes the conjunction operation.

- $S \in N$ is the startsymbol.

A word $w \in \Sigma^*$ is in the language generated by the grammar when a derivation sequence, as defined in Definition 5.2: $S \Rightarrow^* w \,\&...\&\, w$ can be constructed (where S is the startsymbol of the grammar).

For a rule $\mathrm{A} \rightarrow \alpha_1 \,\&\, \alpha_2$, when applied in a word $u\mathrm{A}v$, the derivation sequence is $u\mathrm{A}v \Rightarrow u\alpha_1 v \,\&\, u\alpha_2 v$. The context is denoted by u and v. The resulting word, constructed by the grammar, must be conform both the conjuncts.

Definition 5.2 (*Derivation sequence for CG*)

We extend the definition of the derivation sequence for context-free grammars (Definition 4.2):

We use the notation $w \Rightarrow v_1 \,\&...\&\, v_n$ if $w = x\mathrm{A}y$ and for all i with $1 \leq i \leq n$: $v_i = x\alpha_i y$, for some $x, y, \alpha_i \in (\Sigma \cup N)^*$, $\mathrm{A} \in N$, with production rule $\mathrm{A} \rightarrow \alpha_1 \,\&...\&\, \alpha_n$ in R.

Below follow a couple of examples of conjunctive grammars. These are relatively simple examples, because they use the conjunction on the top level. The examples are provided to show how conjunctive grammars can describe more languages than context-free grammars, while still being very intuitive to understand. Later on, we show some more complicated examples.

Example 5.3

The following conjunctive grammar G_3 generates the language $\mathrm{L}_3 = \{a^n b^n c^n \mid n \geq 0\}$, with $\Sigma = \{a, b, c\}$:

$$\begin{aligned} \mathrm{S} &\rightarrow \mathrm{AB} \,\&\, \mathrm{DC} \\ \mathrm{A} &\rightarrow a\mathrm{A} \mid \varepsilon \\ \mathrm{B} &\rightarrow b\mathrm{B}c \mid \varepsilon \\ \mathrm{C} &\rightarrow c\mathrm{C} \mid \varepsilon \\ \mathrm{D} &\rightarrow a\mathrm{D}b \mid \varepsilon \end{aligned}$$

Note that $\mathrm{L}_3 = \{a^i b^j c^k \mid j = k \wedge i, j, k \geq 0\} \cap \{a^i b^j c^k \mid i = j \wedge i, j, k \geq 0\}$, and it can be shown that AB generates the language $\mathrm{L}_{3.1} = \{a^i b^j c^k \mid j = k \wedge i, j, k \geq 0\}$, and that DC generates the language $\mathrm{L}_{3.2} = \{a^i b^j c^k \mid i = j \wedge i, j, k \geq 0\}$.

Example 5.4

The following conjunctive grammar G_4 generates the language $\mathrm{L}_4 = \{a^n b^m c^n d^m \mid m, n \geq 0\}$, with $\Sigma = \{a, b, c, d\}$:

$$\begin{aligned} \mathrm{S} &\rightarrow a\mathrm{X}c\mathrm{D} \,\&\, \mathrm{A}b\mathrm{Y}d \mid \varepsilon \\ \mathrm{A} &\rightarrow a\mathrm{A} \mid \varepsilon \\ \mathrm{B} &\rightarrow b\mathrm{B} \mid \varepsilon \\ \mathrm{C} &\rightarrow c\mathrm{C} \mid \varepsilon \\ \mathrm{D} &\rightarrow d\mathrm{D} \mid \varepsilon \\ \mathrm{X} &\rightarrow a\mathrm{X}c \mid b\mathrm{B} \\ \mathrm{Y} &\rightarrow b\mathrm{Y}d \mid c\mathrm{C} \end{aligned}$$

Note that $\mathrm{L}_4 = \{a^n b^k c^n d^l \mid n, k, l \geq 0\} \cap \{a^k b^m c^l d^m \mid m, k, l \geq 0\}$, and it can be shown that $a\mathrm{X}c\mathrm{D}$ generates the language $\mathrm{L}_{4.1} = \{a^n b^k c^n d^l \mid n, k, l \geq 0\}$, and that $\mathrm{A}b\mathrm{Y}d$ generates the language $\mathrm{L}_{4.2} = \{a^k b^m c^l d^m \mid m, k, l \geq 0\}$.

Example 5.5

The following conjunctive grammar G_5 generates the language $\mathrm{L}_5 = \{w \mid w \in \{a,b,c\}^*, |w|_a = |w|_b = |w|_c\}$, with $\Sigma = \{a,b,c\}$.:

$$\begin{aligned}
\mathrm{S} &\rightarrow \mathrm{X} \,\&\, \mathrm{Y} \mid \varepsilon \\
\mathrm{X} &\rightarrow a\mathrm{X}b\mathrm{C} \mid b\mathrm{X}a\mathrm{C} \mid \mathrm{C}a\mathrm{X}b \mid \mathrm{C}b\mathrm{X}a \mid c \\
\mathrm{Y} &\rightarrow b\mathrm{Y}c\mathrm{A} \mid c\mathrm{Y}b\mathrm{A} \mid \mathrm{A}b\mathrm{Y}c \mid \mathrm{A}c\mathrm{Y}b \mid a \\
\mathrm{C} &\rightarrow c\mathrm{C} \mid \varepsilon \\
\mathrm{A} &\rightarrow a\mathrm{A} \mid \varepsilon
\end{aligned}$$

Note that
$\mathrm{L}_5 = \{w \mid w \in \{a,b,c\}^*, |w|_a = |w|_b\} \cap \{w \mid w \in \{a,b,c\}^*, |w|_b = |w|_c\}$
and it can be shown that X generates the language
$\mathrm{L}_{5.1} = \{w \mid w \in \{a,b,c\}^*, |w|_a = |w|_b\}$, and that Y generates the language $\mathrm{L}_{5.2} = \{w \mid w \in \{a,b,c\}^*, |w|_b = |w|_c\}$.

5.1.1 Proving that a conjunctive grammar produces a Language

For the conjunctive grammars that we have shown thus far it is relatively easy to understand what language they produce, since all these languages are essentially the intersection of two context-free languages.

Conjunctive grammars also allow to use the conjunction operation more freely. However, this results in the fact that it is a lot harder to see what language is generated by the grammar. An example is the conjunctive grammar G_6, as defined in Theorem 5.6. In this section we will prove that this conjunctive grammar G_6, generates exactly language $\mathrm{L}_6 = \{wcw \mid w \in \{a, b\}^*\}$, with $\Sigma = \{a, b, c\}$.

To prove that a grammar G produces a language L we need to show two things. First, we need to prove that all the words that can be generated by the grammar G, are in the language L. Second, we need to prove that all the words that are in the language L can be generated by the grammar G.

Theorem 5.6 (*CG produces language wcw*)

The following conjunctive grammar G_6 generates the language $\mathrm{L}_6 = \{wcw \mid w \in \{a, b\}^*\}$, with $\Sigma = \{a, b, c\}$:

$$
\begin{aligned}
\mathrm{S} &\rightarrow \mathrm{C} \;\&\; \mathrm{D} \\
\mathrm{C} &\rightarrow \mathrm{XCX} \mid c \\
\mathrm{D} &\rightarrow a\mathrm{A} \;\&\; a\mathrm{D} \mid b\mathrm{B} \;\&\; b\mathrm{D} \mid c\mathrm{E} \\
\mathrm{A} &\rightarrow \mathrm{XAX} \mid c\mathrm{E}a \\
\mathrm{B} &\rightarrow \mathrm{XBX} \mid c\mathrm{E}b \\
\mathrm{E} &\rightarrow \mathrm{XE} \mid \varepsilon \\
\mathrm{X} &\rightarrow a \mid b
\end{aligned}
$$

It has been shown that L_6 is not context-free [20]. The grammar uses the c in the word as center marker and orientation point, and therefore this method cannot be applied to writing a conjunctive grammar for the language $\{ww \mid w \in \{a, b\}^*\}$. It remains an open problem whether conjunctive grammars can generate this latter language [11].

The proof of Theorem 5.6 is split into 4 different parts.

Part 1

Lemma 5.7 (*CFG produces language xcy*)

The following context-free grammar $G_{6.1}$ generates the language $\mathrm{L}_{6.1} = \{xcy \mid x, y \in \{a, b\}^*, |x| = |y|\}$, with $\Sigma = \{a, b, c\}$ and where C is the starting symbol:

$$\begin{aligned} \mathrm{C} &\rightarrow \mathrm{XCX} \mid c \\ \mathrm{X} &\rightarrow a \mid b \end{aligned}$$

Step 1: Induction on the length of the derivation

We first need to prove that all the words that are generated by grammar $G_{6.1}$ are indeed in the language $\mathrm{L}_{6.1}$. We do this by an induction proof on the length of the derivation sequence.

Let $\mathrm{C} \Rightarrow u_1 \Rightarrow ... \Rightarrow u_n$ be a derivation sequence in C, where $u_k \in \{\mathrm{C}, \mathrm{X}, a, b, c\}^*$, for $k = 1, ..., n$.

Let $\mathcal{A} = \{\mathrm{X}, a, b\}^*$
Let $\mathcal{B} = \{\mathrm{C}, c\}$

We define an invariant, the property we will use for our proof:
$\forall(k \in \mathbb{N})\exists(x, y \in \mathcal{A}, z \in \mathcal{B})(u_k = xzy \wedge |x| = |y|)$,

When $u_n \in \{a, b, c\}^*$,
then $z = c$,
with $x, y \in \{a, b\}^*$ and $|x| = |y|$, and therefore $u_n \in \mathrm{L}_{6.1}$.

Base case: length of the derivation is 1

Let $w \in \{a, b, c\}^*$ and $|w| \geq 1$. Suppose $\mathrm{C} \Rightarrow^* w$.

1. The only derivation sequence starting from C of length 1 is $\mathrm{C} \Rightarrow c$.
2. From 1 it follows that $c \in \mathcal{B}$.
3. From 1 and 2 it follows that w is of the form xzy, with $x, y \in \mathcal{A}$, $|x| = |y| = 0$ and $z = c$.

It follows that the property holds for the base case.

Induction step: length of the derivation is $i+1$.
Suppose that $\exists(x, y \in \mathcal{A}, z \in \mathcal{B})(u_i = xzy \wedge |x| = |y|)$ for some i (Induction Hypothesis) and that $u_i \Rightarrow u_{i+1}$.

To prove: $\exists(x, y \in \mathcal{A}, z \in \mathcal{B})(u_{i+1} = xzy \wedge |x| = |y|)$.

There are three cases to be considered.

Case 1: $u_i \Rightarrow u_{i+1}$ is of the form $xzy \Rightarrow x'zy$.
Then one of two production rules was used:

1. Suppose the production rule $\mathrm{X} \to a$ was used.
   (i) Then $x' \in \mathcal{A}$.
   (ii) Then $|x'|_{\mathrm{X}} = |x|_{\mathrm{X}} - 1$ and $|x'|_a = |x|_a + 1$, and thus $|x'| = |x|$.
   (iii) From (ii) it follows that $|x'| = |y|$.
   (iv) From (i), (iii) and the Induction Hypothesis it follows that the property holds when this production rule was used.
2. Suppose the production rule $\mathrm{X} \to b$ was used. This proof is similar to the one above and is therefore ommitted.

The property holds in this case.

Case 2: $u_i \Rightarrow u_{i+1}$ is of the form $xzy \Rightarrow xz'y$.
Then one of two production rules was used:

1. Suppose the production rule $\mathrm{C} \to \mathrm{XCX}$ was used.
   (i) Then $xz'y = x\mathrm{XCX}y$.
   (ii) Then, since $|x| = |y|$, $|x\mathrm{X}| = |\mathrm{X}y|$.
   (iii) Then $\mathrm{C} \in \mathcal{B}$.
   (iv) From (ii), (iii) and the Induction Hypothesis it follows that there exists $x'' = x\mathrm{X}, y'' = \mathrm{X}y, z'' = \mathrm{C}$ such that u_{i+1} is of the form $x''z''y''$, where $x'', y'' \in \mathcal{A}$, and thus the property holds.
2. Suppose the production rule $\mathrm{C} \to c$ was used.
   (i) Then $xz'y = xcy$.
   (ii) It follows that $z' \in \mathcal{B}$.
   (iii) From (i), (ii) and the Induction Hypothesis it follows that the property holds when this production rule is used.

The property holds in this case.

Case 3: $u_i \Rightarrow u_{i+1}$ is of the form $xzy \Rightarrow xzy'$. This case is similar to case 1 and is therefore ommitted.

This concludes step 1 of the proof.

Step 2: Induction on the length of x and y
In this step we want to prove that all the words in language $\mathrm{L}_{6.1}$ can be generated by grammar $G_{6.1}$. We prove this by induction on the length of x and y.

To prove:
For all $w \in \{a, b, \mathrm{C}\}^*$, with $|w| \geq 1$:
If w is of the form $x\mathrm{C}y$, with $x, y \in \{a, b\}^*$ and $|x| = |y|$,
Then $\mathrm{C} \Rightarrow^* w$.

Whenever we show that $\mathrm{C} \Rightarrow^* x\mathrm{C}y$, it implies that $\mathrm{C} \Rightarrow^* xcy$, because of the production rule $\mathrm{C} \rightarrow c$. This simplifies our proof.

Base case: length of x and y is 0
Let $w = x\mathrm{C}y$, with $x, y \in \{a, b\}^*$ and $|x| = |y| = 0$.

1. Then $w = \mathrm{C}$.
2. It is obvious that $\mathrm{C} \Rightarrow^* \mathrm{C}$.
3. From 1 and 2 it follows that the property holds.

Induction step: length of x and y is $i + 1$.
Suppose that it holds that for all $w \in \{a, b, c, \mathrm{C}\}^*$, with $|w| \geq 1$ and w of the form $x\mathrm{C}y$, with $x, y \in \{a, b\}^*$ and $|x| = |y| = i$, we have $\mathrm{C} \Rightarrow^* w$. (Induction Hypothesis).

We want to prove that when we have $w' \in \{a, b, c, \mathrm{C}\}^*$, with $w' = x'\mathrm{C}y'$, where $x, y \in \{a, b\}^*$ $|x'| = |y'| = i + 1$, we can create a derivation sequence $\mathrm{C} \Rightarrow^* x'\mathrm{C}y'$.

1. Let $x' = x\alpha_1$ and $y' = \alpha_2 y$ with $\alpha_1, \alpha_2 \in \{a, b\}$.
2. We know that the nonterminal X generates either an a or a b.
3. From 1, 2 and the Induction Hypothesis it follows that we can construct a derivation sequence $\mathrm{C} \Rightarrow^* x\mathrm{C}y \Rightarrow x\mathrm{XCX}y \Rightarrow^* x\alpha_1\mathrm{C}\alpha_2 y$, which is $x'\mathrm{C}y'$.

This concludes the proof of Lemma 5.7.

Part 2

Lemma 5.8 (*CFG produces language xcvay*)

The following context-free grammar $G_{6.2}$ generates the language $\mathrm{L}_{6.2} = \{xcvay \mid x, v, y \in \{a, b\}^*, |x| = |y|\}$, with $\Sigma = \{a, b, c\}$, where A is the starting symbol:

$$\begin{aligned} \mathrm{A} &\rightarrow \mathrm{XAX} \mid c\mathrm{E}a \\ \mathrm{E} &\rightarrow \mathrm{XE} \mid \varepsilon \\ \mathrm{X} &\rightarrow a \mid b \end{aligned}$$

Step 1: Induction on the length of the derivation

We first need to prove that all the words that are generated by grammar $G_{6.2}$ are indeed in the language $\mathrm{L}_{6.2}$. We do this by an induction proof on the length of the derivation sequence.

Let $\mathrm{A} \Rightarrow u_1 \Rightarrow ... \Rightarrow u_n$ be a derivation in A, where $u_k \in \{\mathrm{A}, \mathrm{E}, \mathrm{X}, a, b, c\}^*$, for $k = 1, ..., n$.

Let $\mathcal{C} = \{\mathrm{X}, a, b\}^*$
Let $\mathcal{D} = \{\mathrm{E}, \mathrm{X}, a, b\}^*$

We define an invariant, the property we will use for our proof:
$\forall(k \in \mathbb{N})\exists(x, y \in \mathcal{C})(u_k = xzy \wedge |x| = |y| \wedge (z = \mathrm{A} \vee \exists(v \in \mathcal{D})(z = cva)))$

When $u_n \in \{a, b, c\}^*$,
then u_n is of the form $xcvay$,
with $x, y, v \in \{a, b\}^*$ and $|x| = |y|$,
and therefore $u_n \in \mathrm{L}_{6.2}$.

Base case: length of the derivation is 2

Let $w \in \{a, b, c\}^*$ and $|w| \geq 2$.
Let $\mathrm{A} \Rightarrow^* w$.

1. Then the derivation of w is $\mathrm{A} \Rightarrow c\mathrm{E}a \Rightarrow ca$.
2. Then w is of the form $xcvay$, with $x, v, y \in \{a, b\}^*$ and $|v| = |x| = |y| = 0$.
3. From 2 it follows that the property holds for the base case.

Induction step: length of the derivation is $i+1$.
Suppose that $\exists(x, y \in \mathcal{C})(u_i = xzy \wedge |x| = |y| \wedge (z = \mathrm{A} \vee \exists(v \in \mathcal{D})(z = cva)))$ for some i (Induction Hypothesis) and that $u_i \Rightarrow u_{i+1}$.

To prove: $\exists(x, y \in \mathcal{C})(u_{i+1} = xzy \wedge |x| = |y| \wedge (z = \mathrm{A} \vee \exists(v \in \mathcal{D})(z = cva)))$.

There are three cases to be considered:

Case 1: $u_i \Rightarrow u_{i+1}$ is of the form $xzy \Rightarrow x'zy$.
Then one of two production rules was used:

1. Suppose the production rule $\mathrm{X} \to a$ was used.
   (i) Then $x' \in \mathcal{C}$.
   (ii) Then $|x'|_{\mathrm{X}} = |x|_{\mathrm{X}} - 1$ and $|x'|_a = |x|_a + 1$, and thus $|x'| = |x|$.
   (iii) From (ii) it follows that $|x'| = |y|$.
   (iv) From the Induction Hypothesis it follows that $z = \mathrm{A} \vee \exists(v \in \mathcal{D})(z = cva)$
   (v) From (i), (iii), (iv) and the Induction Hypothesis it follows that the property holds when this production rule was used.
2. Suppose the production rule $\mathrm{X} \to b$ was used. This proof is similar to the one above and is therefore ommitted.

The property holds in this case.

Case 2: $u_i \Rightarrow u_{i+1}$ is of the form $xzy \Rightarrow xz'y$.
From the Induction Hypothesis we know that $z = \mathrm{A} \vee \exists(v \in \mathcal{D})(z = cva)$.

Case 2.1: $z = \mathrm{A}$
Then one of two production rules was used:

1. Suppose the production rule $\mathrm{A} \to \mathrm{XAX}$ was used.
   (i) The derivation is $x\mathrm{A}y \Rightarrow x\mathrm{XAX}y$
   (ii) From (i) it follows that $|x\mathrm{X}| = |\mathrm{X}y|$.
   (iii) We can then define a $z'' = \mathrm{A}$.
   (iv) From (ii), (iii) and the Induction Hypothesis it follows that there exists $x'' = x\mathrm{X}, y'' = \mathrm{X}y, z'' = \mathrm{A}$ such that the property holds.
2. Suppose the production rule $\mathrm{A} \to c\mathrm{E}a$ was used.
   (i) The derivation is $x\mathrm{A}y = xc\mathrm{E}ay$.
   (ii) We define $v = \mathrm{E}$.

(iii) From (i) and (ii) it follows that $\exists(v \in \mathcal{D})(z' = cva)$.

(iv) From (iii) and the Induction Hypothesis it follows that the property holds when this production rule is used.

The property holds in Case 2.1.

Case 2.2: $\exists(v \in \mathcal{D})(z = cva)$
Then one of two production rules was used:

1. Suppose the production rule $\mathrm{E} \to \mathrm{XE}$ was used.

   (i) The derivation is $xcvay \Rightarrow xcv'ay$

   (ii) Then $v' \in \mathcal{D}$.

   (iii) From (ii) it follows that $\exists(v' \in \mathcal{D})(z' = cv'a)$.

   (iv) From (iii) and the Induction Hypothesis it follows that the property holds.

2. Suppose the production rule $\mathrm{E} \to \varepsilon$ was used.

   (i) The derivation is $xcvay = xcvay$.

   (ii) From (i) and the Induction Hypothesis it follows that the property holds when this production rule is used.

The property holds in Case 2.2, and therefore holds in Case 2.

Case 3: $u_i \Rightarrow u_{i+1}$ is of the form $xzy \Rightarrow xzy'$. This case is similar to case 1 and is therefore ommitted.

This concludes step 1 of the proof.

Step 2
In this step we want to prove that all the words in language $\mathrm{L}_{6.2}$ can be generated by grammar $G_{6.2}$. This can be proven by induction on the length of v and induction on the length of x and y.

However, we can easily see that $\mathrm{E} \Rightarrow^* v$, for all $v \in \{a, b\}^*$.
Furthermore, we observe that $\mathrm{A} \Rightarrow^* \mathrm{X}^n\mathrm{AX}^n$ for all $n \in \mathbb{N}$.
Combining this we get $\mathrm{A} \Rightarrow^* \mathrm{X}^n\mathrm{AX}^n \Rightarrow \mathrm{X}^n c\mathrm{E}a\mathrm{X}^n \Rightarrow^* xcvay$ for any $x, v, y \in \{a, b\}^* \wedge |x| = |y|$. This shows that for every word $w \in \mathrm{L}_{6.2}$, we have $\mathrm{A} \Rightarrow^* w$.

This concludes the proof of Lemma 5.8.

Corollary 5.9 (*CFG produces language xcvby*)

The following context-free grammar $G_{6.3}$ generates the language $\mathrm{L}_{6.3} = \{xcvby \mid x, v, y \in \{a, b\}^*, |x| = |y|\}$, with $\Sigma = \{a, b, c\}$, where B is the starting symbol:

$$\begin{aligned} \mathrm{B} &\rightarrow \mathrm{XBX} \mid c\mathrm{E}b \\ \mathrm{E} &\rightarrow \mathrm{XE} \mid \varepsilon \\ \mathrm{X} &\rightarrow a \mid b \end{aligned}$$

This grammar is similar to the grammar $G_{6.2}$.

Part 3

Lemma 5.10 (*CG produces language uczu*)

The following conjunctive grammar $G_{6.4}$ generates the language $\mathrm{L}_{6.4} = \{uczu \mid u, z \in \{a, b\}^*\}$, where D is the starting symbol:

$$\begin{aligned} \mathrm{D} &\rightarrow a\mathrm{A} \,\&\, a\mathrm{D} \mid b\mathrm{B} \,\&\, b\mathrm{D} \mid c\mathrm{E} \\ \mathrm{A} &\rightarrow \mathrm{XAX} \mid c\mathrm{E}a \\ \mathrm{B} &\rightarrow \mathrm{XBX} \mid c\mathrm{E}b \\ \mathrm{E} &\rightarrow \mathrm{XE} \mid \varepsilon \\ \mathrm{X} &\rightarrow a \mid b \end{aligned}$$

The shortest derivation in D is $\mathrm{D} \Rightarrow c\mathrm{E} \Rightarrow c$. Then we see that c is of the form $uczu$, with $|u| = |z| = 0$. So $c \in \mathrm{L}_{6.4}$.

The longer derivations in D are of the form

$$\mathrm{D} \Rightarrow \alpha_1\beta_1 \,\&\, \alpha_1\mathrm{D} \Rightarrow^* \alpha_1\beta_1 \,\&\ldots\&\, \alpha_1\ldots\alpha_i\beta_i \,\&\ldots\&\, \alpha_1\ldots\alpha_n\beta_n \,\&\, uc\mathrm{E}$$

where $\alpha_k \in \{a, b\} \wedge \beta_k = \begin{cases} \mathrm{A} & \text{if } \alpha_k = a \\ \mathrm{B} & \text{if } \alpha_k = b \end{cases}$
and $u = \alpha_1\ldots\alpha_n \wedge |u| = n$
and where $i, k, n \in \mathbb{N} \wedge 1 \leq i, k \leq n$

A word w generated by D must be representable by every conjunct in the derivation sequence. In this case, the word generated by D must fit in the following forms:

- $\alpha_1\beta_1$, which is a word of the form $\alpha_1 x_1 c v_1 \alpha_1 y_1$ (according to Lemma 5.8 and Corollary 5.9), where $x_1, v_1, y_1 \in \{a, b\}^* \wedge |x_1| = |y_1|$.
- ...

- $\alpha_1...\alpha_i\beta_i$, which is a word of the form $\alpha_1...\alpha_i x_i c v_i \alpha_i y_i$, where $x_i, v_i, y_i \in \{a,b\}^* \wedge |x_i| = |y_i|$.
- ...
- $\alpha_1...\alpha_n\beta_n$, which is a word of the form $\alpha_1...\alpha_n x_n c v_n \alpha_n y_n$, where $x_n, v_n, y_n \in \{a,b\}^* \wedge |x_n| = |y_n| = 0$, so of the form $\alpha_1...\alpha_n c v_n \alpha_n$.
- ucE, which is a word of the form $\alpha_1...\alpha_n ct$, where $t \in \{a,b\}^*$.

This means that w, conform all the conjuncts, is of the form

$$\alpha_1 x_1 c v_1 \alpha_1 y_1 = ... = \alpha_1...\alpha_i x_i c v_i \alpha_i y_i = ... = \alpha_1...\alpha_n c v_n \alpha_n = \alpha_1...\alpha_n ct$$

This leads to the following conclusions:

- $\alpha_1 x_1 = ... = \alpha_1...\alpha_i x_i = ... = \alpha_1...\alpha_n = u$.
- $v_1\alpha_1 y_1 = ... = v_i\alpha_i y_i = ... = v_n\alpha_n = t$.
- t is of the form $v_1\alpha_1...\alpha_n = v_1 u$.

Combining these observations, we conclude that every conjunct is of the form $ucvu$. So whenever D $\Rightarrow^*$ w, w must be of the form $uczu$, with $u, z \in \{a,b\}^*$ which means that $w \in \mathrm{L}_{6.4}$.

The other way around also holds. Suppose that a word $w \in \mathrm{L}_{6.4}$, where $|w| = 1$. Then we can construct the derivation sequence D $\Rightarrow c$E $\Rightarrow c$.

Now suppose we know for some word $w = uzcu \in \mathrm{L}_{6.4}$, with $|u| = i$, it is the case that D $\Rightarrow^*$ $uczu$. This is our Induction Hypothesis.

We need to prove that for a word $w' = u'cz'u'$, with $u', z' \in \{a,b\}^*$ and $|u'| = i + 1$, we can construct a derivation sequence with the grammar.

Let $u' = \alpha u$, with $\alpha \in \{a,b\}$. Then we need to construct a derivation sequence D $\Rightarrow^*$ $\alpha ucz'\alpha u$.

Applying the Induction Hypotheses, where we take $z'\alpha$ as z, we can construct a derivation sequence D $\Rightarrow^*$ $ucz'\alpha u$. Dependent on whether α is an a or a b, we insert the production rule D $\rightarrow a$A & aD or D $\rightarrow b$B & bD. We will only prove one of these cases, as their proof is almost identical.

Suppose $\alpha = a$. We get D $\Rightarrow a$A & aD. We know that we can construct the sequence aD $\Rightarrow^*$ $aucz'au$.

We still need to show that aA $\Rightarrow$ $aucz'au$. From Lemma 5.8 we know

that we can construct the derivation sequence A $\Rightarrow^*$ $xcvay$, where $x, v, y \in \{a, b\}^* \wedge |x| = |y|$. If we take $x = y = u$, and $v = z'$, we can construct the derivation sequence A $\Rightarrow^*$ $ucz'au$.

This means that we can construct a derivation sequence D $\Rightarrow$ aA & aD $\Rightarrow^*$ $aucz'au$ & $aucz'au$, which is what we had to prove.

This concludes the proof of Lemma 5.10.

Part 4

Recall the Theorem:

Theorem 5.6 (*CG produces language wcw*)

The following conjunctive grammar G_6 generates the language $L_6 = \{wcw \mid w \in \{a, b\}^*\}$, with $\Sigma = \{a, b, c\}$:

$$
\begin{aligned}
S &\rightarrow C \,\&\, D \\
C &\rightarrow XCX \mid c \\
D &\rightarrow a\text{A} \,\&\, a\text{D} \mid b\text{B} \,\&\, b\text{D} \mid c\text{E} \\
A &\rightarrow XAX \mid c\text{E}a \\
B &\rightarrow XBX \mid c\text{E}b \\
E &\rightarrow XE \mid \varepsilon \\
X &\rightarrow a \mid b
\end{aligned}
$$

Every word generated by this grammar is a word that occurs both in $L_{6.1}$ and in $L_{6.4}$. These are words of the form xcy, with $x, y \in \{a, b\}^* \wedge |x| = |y|$, and words of the form $uczu$, with $u, z \in \{a, b\}^*$. So when $xcy = uczu$, it must be the case that $x = u$, and $y = zu$, and therefore, $|z| = 0$. These are words that are of the form wcw, which is exactly what the Theorem states. This concludes the proof of Theorem 5.6.

5.2 Conjunctive grammars in terms of a Formal Deduction System

Similar to context-free grammars, we can define the meaning of conjunctive grammars in terms of a formal deduction system.

Definition 5.11 (*Formal deduction system for conjunctive grammars [14]*)

For a conjunctive grammar $G = (\Sigma, N, R, S)$ a formal deduction system $\vdash_G$ can be defined as follows. Production rules that don't use the conjunction operation are defined the same as the production rules in context-free grammars (see Definition 4.4). For every production rule $\mathrm{A} \to \alpha_1$ &...& α_n in G, there is a corresponding inference rule in $\vdash_G$, where $w \in \Sigma^*$:

$$\frac{\alpha_1(w) \quad ... \quad \alpha_n(w)}{\mathrm{A}(w)} \; [rulename]$$

A word w is in the language generated by the grammar when an inference tree can be constructed with $S(w)$ as its conclusion (where S is the startsymbol of the grammar).

In Section 4.2 we observed that in context-free grammars, applying an inference rule on some term results in "slicing" the term. Take rule $[\mathrm{A}^1]$ of Table 5.1: when applying this rule to word w, it results in three terms above the line: u_1, u_2 and u_3. All these branches take a part of w, and they do not overlap.

But in conjunctive grammars there is another possibility. For example, applying rule $[\mathrm{S}^1]$ of Table 5.1 to a word w results in two branches that take the same term. This is because of the conjunction. In order to prove that word w can be produced by S, it needs to be shown that the word can be generated by both the nonterminals H and Y.

One might think that this means that it is harder to parse conjunctive grammars. However, the difficult part of parsing the grammars is no different than in context-free grammars: finding the right position to cut the term. Although the inference trees for conjunctive grammars are likely to have more branches than context-free grammars, the upper bounds on the time complexity of parsing algorithms remain the same, as stated by Okhotin [11].

Below we sketch an example of a conjunctive grammar, first in terms of rewriting, followed by corresponding formal deduction system. Furthermore, an inference tree is included to show how the inference rules can be applied to a term.

Example 5.12

The following conjunctive grammar G_7 generates the language $\mathrm{L}_7 = \{(wc)^{|w|} \mid w \in \{a,b\}^*\}$, with $\Sigma = \{a,b,c\}$:

$$
\begin{aligned}
\mathrm{S} &\to \mathrm{H}\ \&\ \mathrm{Y} \mid ac \mid bc \mid \varepsilon \\
\mathrm{H} &\to a\mathrm{HG} \mid b\mathrm{HG} \mid ac \mid bc \\
\mathrm{G} &\to a\mathrm{G} \mid b\mathrm{G} \mid c \\
\mathrm{Y} &\to \mathrm{YG}\ \&\ \mathrm{GY} \mid \mathrm{F}c \\
\mathrm{F} &\to \mathrm{C}\ \&\ \mathrm{D} \\
\mathrm{C} &\to \mathrm{XCX} \mid c \\
\mathrm{D} &\to a\mathrm{A}\ \&\ a\mathrm{D} \mid b\mathrm{B}\ \&\ b\mathrm{D} \mid c\mathrm{E} \\
\mathrm{A} &\to \mathrm{XAX} \mid c\mathrm{E}a \\
\mathrm{B} &\to \mathrm{XBX} \mid c\mathrm{E}b \\
\mathrm{E} &\to \mathrm{XE} \mid \varepsilon \\
\mathrm{X} &\to a \mid b
\end{aligned}
$$

Language L_7 uses the language $\mathrm{L}_6 = \{wcw \mid w \in \{a,b\}^*\}$, which is represented by the nonterminal F.

Table 5.1 lists the inference rules for grammar $\vdash_{G_7}$.

Table 5.1: The inference rules for the conjunctive grammar $\vdash_{G_7}$

Rule		Rule name
$\vdash a(a)$		[axiom^a]
$\vdash b(b)$		[axiom^b]
$\vdash c(c)$		[axiom^c]
$\vdash \mathrm{S}(\varepsilon)$		[S^0]
$\vdash \mathrm{E}(\varepsilon)$		[E^0]
$\dfrac{\mathrm{H}(w) \qquad \mathrm{Y}(w)}{\mathrm{S}(w)}$		[S^1]
$\dfrac{a(a) \qquad c(c)}{\mathrm{S}(ac)}$		[S^2]
$\dfrac{b(b) \qquad c(c)}{\mathrm{S}(bc)}$		[S^3]
$\dfrac{a(a) \qquad \mathrm{H}(u_1) \qquad \mathrm{G}(u_2)}{\mathrm{H}(aw)}$	where $w = u_1 u_2$	[H^1]

Table 5.1 (continued)

Rule		Rule name
$\dfrac{b(b) \quad \mathrm{H}(u_1) \quad \mathrm{G}(u_2)}{\mathrm{H}(bw)}$	where $w = u_1u_2$	$[\mathrm{H}^2]$
$\dfrac{a(a) \quad c(c)}{\mathrm{H}(ac)}$		$[\mathrm{H}^3]$
$\dfrac{b(b) \quad c(c)}{\mathrm{H}(bc)}$		$[\mathrm{H}^4]$
$\dfrac{a(a) \quad \mathrm{G}(w)}{\mathrm{G}(aw)}$		$[\mathrm{G}^1]$
$\dfrac{b(b) \quad \mathrm{G}(w)}{\mathrm{G}(bw)}$		$[\mathrm{G}^2]$
$\dfrac{c(c)}{\mathrm{G}(c)}$		$[\mathrm{G}^3]$
$\dfrac{\mathrm{Y}(u_1) \quad \mathrm{G}(u_2) \quad \mathrm{G}(v_1) \quad \mathrm{Y}(v_2)}{\mathrm{Y}(w)}$	where $w = u_1u_2 = v_1v_2$	$[\mathrm{Y}^1]$
$\dfrac{\mathrm{F}(w) \quad c(c)}{\mathrm{Y}(wc)}$		$[\mathrm{Y}^2]$
$\dfrac{\mathrm{C}(w) \quad \mathrm{D}(w)}{\mathrm{F}(w)}$		$[\mathrm{F}^1]$
$\dfrac{\mathrm{X}(u_1) \quad \mathrm{C}(u_2) \quad \mathrm{X}(u_3)}{\mathrm{C}(w)}$	where $w = u_1u_2u_3$	$[\mathrm{C}^1]$
$\dfrac{c(c)}{\mathrm{C}(c)}$		$[\mathrm{C}^2]$
$\dfrac{a(a) \quad \mathrm{A}(w) \quad a(a) \quad \mathrm{D}(w)}{\mathrm{D}(aw)}$		$[\mathrm{D}^1]$
$\dfrac{b(b) \quad \mathrm{B}(w) \quad b(b) \quad \mathrm{D}(w)}{\mathrm{D}(bw)}$		$[\mathrm{D}^2]$
$\dfrac{c(c) \quad \mathrm{E}(w)}{\mathrm{D}(cw)}$		$[\mathrm{D}^3]$
$\dfrac{\mathrm{X}(u_1) \quad \mathrm{A}(u_2) \quad \mathrm{X}(u_3)}{\mathrm{A}(w)}$	where $w = u_1u_2u_3$	$[\mathrm{A}^1]$
$\dfrac{c(c) \quad \mathrm{E}(w) \quad a(a)}{\mathrm{A}(cwa)}$		$[\mathrm{A}^2]$
$\dfrac{\mathrm{X}(u_1) \quad \mathrm{B}(u_2) \quad \mathrm{X}(u_3)}{\mathrm{B}(w)}$	where $w = u_1u_2u_3$	$[\mathrm{B}^1]$
$\dfrac{c(c) \quad \mathrm{E}(w) \quad b(b)}{\mathrm{B}(cwb)}$		$[\mathrm{B}^2]$

Table 5.1 (continued)

Rule		Rule name
$\dfrac{\mathrm{X}(u_1) \quad \mathrm{E}(u_2)}{\mathrm{E}(w)}$	where $w = u_1 u_2$	[E^1]
$\dfrac{a(a)}{\mathrm{X}(a)}$		[X^1]
$\dfrac{b(b)}{\mathrm{X}(b)}$		[X^2]

We construct a proof tree to show that the word *bbacbbacbbac* is in the language L_7:

$$\dfrac{b(b) \quad \dfrac{\dfrac{a(a) \quad c(c)}{\mathrm{H}(ac)}\,[\mathrm{H}^3] \quad \dfrac{\text{SUBTREE 1}}{\mathrm{G}(bbac)}}{\mathrm{H}(bacbbac)}\,[\mathrm{H}^2] \quad \dfrac{\text{SUBTREE 1}}{\mathrm{G}(bbac)}}{\mathrm{H}(bbacbbacbbac)}\,[\mathrm{H}^2] \quad \dfrac{\text{SUBTREE 2}}{\mathrm{Y}(bbacbbacbbac)}}{\mathrm{S}(bbacbbacbbac)}\,[\mathrm{S}^1]$$

SUBTREE 1:

$$\dfrac{b(b) \quad \dfrac{\dfrac{a(a) \quad \dfrac{c(c)}{\mathrm{G}(c)}\,[\mathrm{G}^3]}{\mathrm{G}(ac)}\,[\mathrm{G}^1]}{\mathrm{G}(bac)}\,[\mathrm{G}^2]}{\mathrm{G}(bbac)}\,[\mathrm{G}^2]$$

SUBTREE 2:

$$\dfrac{\text{SUBTREE 3} \quad \text{SUBTREE 1} \quad \text{SUBTREE 1} \quad \text{SUBTREE 3}}{\mathrm{Y}(bbacbbacbbac)}\,[\mathrm{Y}^1]$$

SUBTREE 3:

$$\dfrac{\dfrac{b(b)}{\mathrm{X}(b)} \quad \dfrac{\dfrac{b(b)}{\mathrm{X}(b)} \quad \dfrac{a(a)}{\mathrm{X}(a)} \quad \dfrac{c(c)}{\mathrm{C}(c)} \quad \dfrac{b(b)}{\mathrm{X}(b)}}{\mathrm{C}(acb)} \quad \dfrac{b(b)}{\mathrm{X}(b)}}{\mathrm{C}(bacbb)} \quad \dfrac{a(a)}{\mathrm{X}(a)}}{\mathrm{C}(bbacbba)}\,[\mathrm{C}^1] \quad \dfrac{\text{SUBTREE 4}}{\mathrm{D}(bbacbba)}}{\mathrm{F}(bbacbba)}\,[\mathrm{F}^1] \quad c(c)}{\mathrm{Y}(bbacbbac)}\,[\mathrm{Y}^2]$$

In SUBTREE 3, the [C^1] rule is used multiple times in a row. The rule name was ommitted for a clearer overview. The rule names [X^1], [X^2] and [C^2] are left out as well.

SUBTREE 4:

$$\dfrac{b(b) \quad \dfrac{\dfrac{a(a)}{\mathrm{X}(a)} \quad \dfrac{c(c) \quad \mathrm{E}(\varepsilon) \quad b(b)}{\mathrm{B}(cb)} \quad \dfrac{b(b)}{\mathrm{X}(b)}}{\mathrm{B}(acbb)} \quad \dfrac{a(a)}{\mathrm{X}(a)}}{\mathrm{B}(bacbba)}\,[\mathrm{B}^1] \qquad b(b) \quad \dfrac{\text{SUBTREE 5}}{\mathrm{D}(bacbba)}}{\mathrm{D}(bbacbba)}\,[\mathrm{D}^2]$$

In SUBTREE 4 the [B^1] rule is used multiple times in a row. The rule name is ommitted from the tree. The rule names [X^1], [X^2] and [B^2] are left out as well. This also holds for SUBTREE 5.

SUBTREE 5:

$$\dfrac{b(b) \quad \dfrac{\dfrac{a(a)}{\mathrm{X}(a)} \quad \dfrac{c(c) \quad \dfrac{\dfrac{b(b)}{\mathrm{X}(b)} \quad \mathrm{E}(\varepsilon)}{\mathrm{E}(b)}\,[\mathrm{E}^1] \quad b(b)}{\mathrm{B}(cbb)} \quad \dfrac{a(a)}{\mathrm{X}(a)}}{\mathrm{B}(acbba)}\,[\mathrm{B}^1] \qquad b(b) \quad \dfrac{\text{SUBTREE 6}}{\mathrm{D}(acbba)}}{\mathrm{D}(bacbba)}\,[\mathrm{D}^2]$$

SUBTREE 6:

$$\dfrac{a(a) \quad \dfrac{c(c) \quad \dfrac{\dfrac{b(b)}{\mathrm{X}(b)} \quad \dfrac{\dfrac{b(b)}{\mathrm{X}(b)} \quad \mathrm{E}(\varepsilon)}{\mathrm{E}(b)}}{\mathrm{E}(bb)} \quad a(a)}{\mathrm{A}(cbba)}\,[\mathrm{A}^2] \qquad a(a) \quad \dfrac{c(c) \quad \dfrac{\dfrac{b(b)}{\mathrm{X}(b)} \quad \dfrac{\dfrac{a(a)}{\mathrm{X}(a)} \quad \mathrm{E}(\varepsilon)}{\mathrm{E}(a)}}{\mathrm{E}(ba)}}{\mathrm{D}(cbba)}\,[\mathrm{D}^3]}{\mathrm{D}(acbba)}\,[\mathrm{D}^1]$$

In SUBTREE 6 the rule names [E^1], [X^1] and [X^2] are ommitted, to ensure a better overview of the tree itself.

5.3 Conjunctive grammars in terms of Language Equations

Definition 5.13 (*Language equations for conjunctive grammars [13]*)

For every conjunctive grammar $G = (\Sigma, N, R, S)$ a system of equations with languages as unknowns $\mathcal{E}_G$ can be defined as follows. We write

$$\mathrm{A} = \bigcup_{\mathrm{A} \to \alpha_1 \,\&\ldots\&\, \alpha_m \in R} \bigcap_{i=1}^{m} \alpha_i$$

for all nonterminals $\mathrm{A} \in N$ as unknown languages, and for all $a \in \Sigma \cup \{\varepsilon\}, a$ denotes $\{a\}$.

Such a system has a least solution (Corollary 5.19). A word $w \in \Sigma^*$ is in the language generated by the grammar when it is in the least solution (for the startsymbol S).

In comparison to Definition 4.10 an intersection is added to the formula, to include the conjunction operation.

Example 5.14

Recall the conjunctive grammar G_3, that generates the language $\mathrm{L}_3 = \{a^n b^n c^n \mid n \geq 0\}$, with $\Sigma = \{a, b, c\}$:

$$\begin{aligned}
\mathrm{S} &\to \mathrm{AB} \,\&\, \mathrm{DC} \\
\mathrm{A} &\to a\mathrm{A} \mid \varepsilon \\
\mathrm{B} &\to b\mathrm{B}c \mid \varepsilon \\
\mathrm{C} &\to c\mathrm{C} \mid \varepsilon \\
\mathrm{D} &\to a\mathrm{D}b \mid \varepsilon
\end{aligned}$$

The corresponding system of language equations $\mathcal{E}_{G_3}$ is:

$$\begin{cases}
\mathrm{S} = \mathrm{AB} \cap \mathrm{DC} \\
\mathrm{A} = \{a\}\mathrm{A} \cup \{\varepsilon\} \\
\mathrm{B} = \{b\}\mathrm{B}\{c\} \cup \{\varepsilon\} \\
\mathrm{C} = \{c\}\mathrm{C} \cup \{\varepsilon\} \\
\mathrm{D} = \{a\}\mathrm{D}\{b\} \cup \{\varepsilon\}
\end{cases}$$

We consider the function $F : (\mathcal{P}(\Sigma^*))^5 \to (\mathcal{P}(\Sigma^*))^5$, derived from the grammar $\mathcal{E}_{G_3}$:

$$F(\mathrm{S},\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}) = (\mathrm{AB} \cap \mathrm{DC}, \{a\}\mathrm{A} \cup \{\varepsilon\}, \{b\}\mathrm{B}\{c\} \cup \{\varepsilon\}, \{c\}\mathrm{C} \cup \{\varepsilon\}, \{a\}\mathrm{D}\{b\} \cup \{\varepsilon\})$$

Iterating with the emptyset gives:

- $F^0(\emptyset,\emptyset,\emptyset,\emptyset,\emptyset) = (\emptyset,\emptyset,\emptyset,\emptyset,\emptyset)$
- $F^1(\emptyset,\emptyset,\emptyset,\emptyset,\emptyset) = (\emptyset,\{\varepsilon\},\{\varepsilon\},\{\varepsilon\},\{\varepsilon\})$
- $F^2(\emptyset,\emptyset,\emptyset,\emptyset,\emptyset) = (\{\varepsilon\},\{\varepsilon,a\},\{\varepsilon,bc\},\{\varepsilon,c\},\{\varepsilon,ab\})$
- $F^3(\emptyset,\emptyset,\emptyset,\emptyset,\emptyset) = (\{\varepsilon,abc\},\{\varepsilon,a,aa\},\{\varepsilon,bc,bbcc\},\{\varepsilon,c,cc\},\{\varepsilon,ab,aabb\})$
- $F^4(\emptyset,\emptyset,\emptyset,\emptyset,\emptyset) = (\{\varepsilon,abc,aabbcc\},\{\varepsilon,a,aa,aaa\},\{\varepsilon,bc,bbcc,bbbccc\},\{\varepsilon,c,cc,ccc\},\{\varepsilon,ab,aabb,aaabbb\})$
- ...

Lemma 5.15 (*CG - Monotonically increasing function*)

Let $F : (\mathcal{P}(\Sigma^*))^5 \to (\mathcal{P}(\Sigma^*))^5$ be the function with $\Sigma = \{a,b,c\}$ and $F(\mathrm{S},\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}) = (\mathrm{AB} \cap \mathrm{DC}, \{a\}\mathrm{A} \cup \{\varepsilon\}, \{b\}\mathrm{B}\{c\} \cup \{\varepsilon\}, \{c\}\mathrm{C} \cup \{\varepsilon\}, \{a\}\mathrm{D}\{b\} \cup \{\varepsilon\})$
F is a monotonically increasing function.

Proof:
Let $(\mathrm{S},\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}) \subseteq (\mathrm{S}',\mathrm{A}',\mathrm{B}',\mathrm{C}',\mathrm{D}')$. Then we need to show that $F(\mathrm{S},\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}) \subseteq F(\mathrm{S}',\mathrm{A}',\mathrm{B}',\mathrm{C}',\mathrm{D}')$.

We have $F(\mathrm{S},\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}) = (\mathrm{AB} \cap \mathrm{DC}, \{a\}\mathrm{A} \cup \{\varepsilon\}, \{b\}\mathrm{B}\{c\} \cup \{\varepsilon\}, \{c\}\mathrm{C} \cup \{\varepsilon\}, \{a\}\mathrm{D}\{b\} \cup \{\varepsilon\})$
and $F(\mathrm{S}',\mathrm{A}',\mathrm{B}',\mathrm{C}',\mathrm{D}') = (\mathrm{A}'\mathrm{B}' \cap \mathrm{D}'\mathrm{C}', \{a\}\mathrm{A}' \cup \{\varepsilon\}, \{b\}\mathrm{B}'\{c\} \cup \{\varepsilon\}, \{c\}\mathrm{C}' \cup \{\varepsilon\}, \{a\}\mathrm{D}'\{b\} \cup \{\varepsilon\})$

But since
$$\begin{array}{lcl} \mathrm{AB} \cap \mathrm{DC} & \subseteq & \mathrm{A}'\mathrm{B}' \cap \mathrm{D}'\mathrm{C}' \\ \{a\}\mathrm{A} \cup \{\varepsilon\} & \subseteq & \{a\}\mathrm{A}' \cup \{\varepsilon\} \\ \{b\}\mathrm{B}\{c\} \cup \{\varepsilon\} & \subseteq & \{b\}\mathrm{B}'\{c\} \cup \{\varepsilon\} \\ \{c\}\mathrm{C} \cup \{\varepsilon\} & \subseteq & \{c\}\mathrm{C}' \cup \{\varepsilon\} \\ \{a\}\mathrm{D}\{b\} \cup \{\varepsilon\} & \subseteq & \{a\}\mathrm{D}'\{b\} \cup \{\varepsilon\} \end{array}$$

We observe that $F(\mathrm{S},\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}) \subseteq F(\mathrm{S}',\mathrm{A}',\mathrm{B}',\mathrm{C}',\mathrm{D}')$. This proves Lemma 5.15.

Proposition 4.14 states that the function over the language equations of context-free grammars is Scott continuous. In the conjunctive grammars intersection is added. In order to apply Kleene's Fixed Point Theorem, we need to show that this operation is also Scott continuous.

Proposition 5.16 (*Scott continuous function of several variables [17]*)

For a partially ordered set $(\mathcal{P}(\Sigma^*), \subseteq)$ that is a complete lattice, function $F : \mathcal{P}(\Sigma^*)^n \to \mathcal{P}(\Sigma^*)^n$ is a Scott continuous function if and only if function $F_i : \mathcal{P}(\Sigma^*)^n \to \mathcal{P}(\Sigma^*)$ is Scott continuous for every i, where $F(X_1, ..., X_n) = (F_1(X_1, ..., X_n), ..., F_n(X_1, ..., X_n))$.

Lemma 5.17 (*CG - Scott continuous function*)

Let $F : (\mathcal{P}(\Sigma^*))^5 \to (\mathcal{P}(\Sigma^*))^5$ be the function with $\Sigma = \{a, b, c\}$ and
$F(\mathrm{S}, \mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}) = (\mathrm{AB} \cap \mathrm{DC}, \{a\}\mathrm{A} \cup \{\varepsilon\}, \{b\}\mathrm{B}\{c\} \cup \{\varepsilon\},$
$\{c\}\mathrm{C} \cup \{\varepsilon\}, \{a\}\mathrm{D}\{b\} \cup \{\varepsilon\})$
F is Scott continuous.

Proof:
The nonterminals $\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}$ only use Scott continuous operations, so according to Proposition 5.16 we only need to show that the function $F : \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Sigma^*)$ with $F(\mathrm{S}) = \mathrm{AB} \cap \mathrm{DC}$ is Scott continuous.

We consider the chain
$F^0(\emptyset) \subseteq F^1(\emptyset) \subseteq F^2(\emptyset) \subseteq F^3(\emptyset) \subseteq F^4(\emptyset) \subseteq ... = \emptyset \subseteq \emptyset \subseteq \{\varepsilon\} \subseteq \{\varepsilon, abc\} \subseteq \{\varepsilon, abc, aabbcc\} \subseteq ...$
If F is Scott continuous, we have $F(\bigcup_i F^i(\emptyset)) = \bigcup_i F(F^i(\emptyset))$.

$$\begin{aligned} F(\bigcup_i F^i(\emptyset)) &= F(\emptyset \cup \emptyset \cup \{\varepsilon\} \cup \{\varepsilon, abc\} \cup \{\varepsilon, abc, aabbcc\} \cup ...) \\ &= F(\{a^n b^n c^n \mid n \geq 0\}) \\ &= (\{a\}\mathrm{A} \cup \{\varepsilon\}(\{b\}\mathrm{B}\{c\} \cup \{\varepsilon\}) \cap (\{c\}\mathrm{C} \cup \{\varepsilon\})(\{a\}\mathrm{D}\{b\} \cup \{\varepsilon\}) \\ &= a^*\{b^m c^m \mid m \geq 0\} \cap \{a^m b^m \mid m \geq 0\}c^* \\ &= \{a^n b^n c^n \mid n \geq 0\} \end{aligned}$$

$$\begin{aligned} \bigcup_i F(F^i(\emptyset)) &= F(\emptyset) \cup F(\emptyset) \cup F(\{\varepsilon\}) \cup F(\{\varepsilon, abc\}) \cup F(\{\varepsilon, abc, aabbcc\}) \cup ... \\ &= \emptyset \cup \{\varepsilon\} \cup \{\varepsilon, abc\} \cup \{\varepsilon, abc, aabbcc\} \cup \{\varepsilon, abc, aabbcc, aaabbbccc\} \cup ... \\ &= \{a^n b^n c^n \mid n \geq 0\} \end{aligned}$$

This proves Lemma 5.17.

Proposition 5.18 (*Intersection operation is Scott continuous [17]*)

The intersection operation, used in conjunctive grammars, is Scott continuous in the partial ordering $(\mathcal{P}(\Sigma^*), \subseteq)$.

Corollary 5.19 (*CG - Least solution*)

Since $(\mathcal{P}(\Sigma^*), \subseteq)$ is a partially ordered set and a complete lattice, we have for every conjunctive grammar $\mathcal{E}_G$ that the *least solution* of the language equations is $\bigcup_{k\geq 0} F^k(\emptyset)$, with the Scott continuous function $F : \mathcal{P}(\Sigma^*)^{|N|} \to \mathcal{P}(\Sigma^*)^{|N|}$ over the nonterminals N of G, derived from the system of equations $\mathcal{E}_G$. The least solution consists of exactly the languages generated by the nonterminals.

Example 5.20

The solution for the grammar $\mathcal{E}_{G_3}$ is $\bigcup_{i\geq 0} F^i(\emptyset) = (\mathrm{S}, \mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D})$ with
$\mathrm{S} = \{a^n b^n c^n \mid n \geq 0\}$,
$\mathrm{A} = a^*$,
$\mathrm{B} = \{b^m c^m \mid m \geq 0\}$,
$\mathrm{C} = c^*$,
$\mathrm{D} = \{a^m b^m \mid m \geq 0\}$
as shown below:

$$
\begin{aligned}
F(\mathrm{S}, \mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}) &= (\mathrm{AB} \cap \mathrm{DC}, \\
&\quad \{a\}\mathrm{A} \cup \{\varepsilon\}, \\
&\quad \{b\}\mathrm{B}\{c\} \cup \{\varepsilon\}, \\
&\quad \{c\}\mathrm{C} \cup \{\varepsilon\}, \\
&\quad \{a\}\mathrm{D}\{b\} \cup \{\varepsilon\}) \\
&= (a^*\{b^m c^m \mid m \geq 0\} \cap \{a^m b^m \mid m \geq 0\}c^*, \\
&\quad \{a\}a^* \cup \{\varepsilon\}, \\
&\quad \{b\}\{b^m c^m \mid m \geq 0\}\{c\} \cup \{\varepsilon\}, \\
&\quad \{c\}c^* \cup \{\varepsilon\}, \\
&\quad \{a\}\{a^m b^m \mid m \geq 0\}\{b\} \cup \{\varepsilon\}) \\
&= (\{a^n b^n c^n \mid n \geq 0\}, \\
&\quad a^*, \\
&\quad \{b^m c^m \mid m \geq 0\}, \\
&\quad c^*, \\
&\quad \{a^m b^m \mid m \geq 0\}) \\
&= (\mathrm{S}, \mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D})
\end{aligned}
$$

In conclusion, it is a fixed point of F and it can be shown that it is a least fixed point of F, and therefore the least solution of $\mathcal{E}_{G_3}$.

Chapter 6

Boolean Grammars

Boolean grammars were introduced by Okhotin in 2004 [10]. They extend the conjunctive grammars by adding the possibility to use the negation operation, represented by complementation. With this extension, some complicated languages can be defined more easily, since it is possible to introduce a production rule that ensures that a term complies with a property, and explicitly not comply with another property. It remains an open problem whether boolean grammars represent more languages than conjunctive grammars [11].

An example is the language $\{a^m b^n c^n \mid m, n \geq 0, m \neq n\}$ over input alphabet $\Sigma = \{a, b, c\}$. The language can be defined by taking the intersection of the context-free language $\{a^i b^j c^k \mid j = k\}$ with the complement of the context-free language $\{a^i b^j c^k \mid i = j\}$.

This chapter elaborates on the definition of boolean grammars as given by Okhotin [10]. We will show that it is difficult to give a sound definition of boolean grammars in terms of rewriting in Section 6.1, since we want to avoid the possibilty to define grammars that logically contradict themselves. In Section 6.2 we discuss a sound definition in terms of language equations. Finally, we will discuss some limitations of this definition.

6.1 Boolean grammars in terms of Rewriting

Where a conjunctive grammar is a context-free grammar with the conjunction operation added, a boolean grammar is a conjunctive grammar with yet another operation included: negation, represented by the complement operation. With this extension, complex languages can be defined more easily. It is possible to define a language where every word in that language complies with rule A, and explicitly not complies with rule B.

Definition 6.1 (*Boolean grammars [11]*)

A *boolean grammar* is a quadruple $G = (\Sigma, N, R, S)$ where:

- Σ is the alphabet.
- N is a finite set of nonterminal symbols.
- R is a finite set of grammar rules of the form

$$\mathrm{A} \rightarrow \alpha_1 \,\&\, \alpha_2 \,\&\, ... \,\&\, \alpha_n \,\&\, \neg\beta_1 \,\&\, \neg\beta_2 \,\&\, ... \,\&\, \neg\beta_m$$

  where $\mathrm{A} \in N,\ n, m \geq 0, m + n \geq 1$ and $\alpha_i, \beta_j \in (\Sigma \cup N)^*$. A conjunct α_i is called a *positive* conjunct, and a conjunct β_j is called a *negative* conjunct.
- $S \in N$ is the startsymbol.

Definition 6.2 (*Derivation sequence for BG*)

We extend the definition of the derivation sequence for conjunctive grammars (Definition 5.2):

We use the notation $w \Rightarrow v_1 \,\&...\&\, v_n \,\&\, \neg u_1 \,\&...\&\, \neg u_m$ if $w = x\mathrm{A}y$, and for all i with $1 \leq i \leq n$: $v_i = x\alpha_i y$, and for all j with $1 \leq j \leq m$: $u_j = x\beta_j y$, for some $x, y, \alpha_i, \beta_j \in (\Sigma \cup N)^*$, $\mathrm{A} \in N$, with production rule $\mathrm{A} \rightarrow \alpha_1 \,\&...\&\, \alpha_n \,\&\, \neg\beta_1 \,\&...\&\, \neg\beta_m$ in R.

Definition 6.3 (*The language generated by a BG - Intuitive definition*)

Intuitively, we define a word $w \in \Sigma^*$ to be in the language generated by the grammar when for all $v_1, ..., v_m$, if

$$S \Rightarrow^* w \,\&...\&\, w \,\&\, \neg v_1 \,\&...\&\, \neg v_m$$

then $v_i \neq w$, with $1 \leq i \leq m$.

Definition 6.3 can lead to a logical contradiction, since it is possible to define a recursive rule where nonterminals depend on each other in a circular way, while using negation. We will show this shortcoming later in Example 6.6. For now, we can illustrate what the intention is of boolean grammars with some examples.

Example 6.4

The following boolean grammar G_8 generates the language $\mathrm{L}_8 = \{a^m b^n c^n \mid m, n \geq 0, m \neq n\}$, with $\Sigma = \{a, b, c\}$:

$$\begin{aligned} \mathrm{S} &\rightarrow \mathrm{AB} \,\&\, \neg\mathrm{DC} \\ \mathrm{A} &\rightarrow a\mathrm{A} \mid \varepsilon \\ \mathrm{B} &\rightarrow b\mathrm{B}c \mid \varepsilon \\ \mathrm{C} &\rightarrow c\mathrm{C} \mid \varepsilon \\ \mathrm{D} &\rightarrow a\mathrm{D}b \mid \varepsilon \end{aligned}$$

Note that $\mathrm{L}_8 = \{a^m b^n c^n \mid m, n \geq 0, m \neq n\} = \{a^i b^j c^k \mid j = k \wedge i \neq j\} = \{a^i b^j c^k \mid j = k\} \cap \overline{\{a^i b^j c^k \mid i = j\}}$, and it can be shown that AB generates the language $\mathrm{L}_{8.1} = \{a^i b^j c^k \mid j = k\}$, and that DC generates the language $\mathrm{L}_{8.2} = \{a^i b^j c^k \mid i = j\}$.

Example 6.5

The following boolean grammar G_9 generates the language $\mathrm{L}_9 = \{ww \mid w \in \{a, b\}^*\}$, with $\Sigma = \{a, b\}$:

$$\begin{aligned} \mathrm{S} &\rightarrow \neg\mathrm{AB} \,\&\, \neg\mathrm{BA} \,\&\, \mathrm{C} \\ \mathrm{A} &\rightarrow \mathrm{XAX} \mid a \\ \mathrm{B} &\rightarrow \mathrm{XBX} \mid b \\ \mathrm{C} &\rightarrow \mathrm{XXC} \mid \varepsilon \\ \mathrm{X} &\rightarrow a \mid b \end{aligned}$$

It can be shown that the nonterminal A generates the context-free language $\mathrm{L}_{9.1} = \{uav \mid u, v \in \{a, b\}^*, |u| = |v|\}$, and that the nonterminal B generates the context-free language $\mathrm{L}_{9.2} = \{ubv \mid u, v \in \{a, b\}^*, |u| = |v|\}$.

Combined, we observe that AB generates the context-free language $\mathrm{L}_{9.3} = \{uavxby \mid u, v, x, y \in \{a, b\}^*, |u| = |x|, |v| = |y|\}$. It contains all strings of even length, where on the left side of the word there is an a at a position where on the right side of the word there is a b at the same position (when cutting the word in half).

The context-free language generated by BA is similar:

$L_{9.4} = \{ubvxay \mid u, v, x, y \in \{a, b\}^*, |u| = |x|, |v| = |y|\}$. It contains all the strings of even length where there is a b on the left side of the word at some position, and on the right side there is an a at that same position.

The nonterminal C generates the context-free language
$L_{9.5} = \{xy \mid x, y \in \{a, b\}^*, |x| = |y|\}$. This language contains all strings of even length.

Language L_9 defines the set of strings of even length without any mismatches:
$L_9 = \overline{L_{9.3}} \cap \overline{L_{9.4}} \cap L_{9.5} = \{ww \mid w \in \{a, b\}^*\}$

Although boolean grammars allow us to define languages that are "context-free" (in the sense that the context is not important for applying a rule), Definition 6.3 is not a waterproof definition. Consider Example 6.6:

Example 6.6

Let G_{10} be a boolean grammar, defined in terms of rewriting:

$$S \rightarrow \neg S$$

Suppose that we want to know for some word w whether $w \in S$. The grammar states that $w \in S$, whenever $w \notin S$. This obviously is a contradiction. We do not want to allow grammars like these. This calls for a stricter definition of boolean grammars.

Definition 6.3 shows the intention of the extension to conjunctive grammars, but should not be used as a formal definition. With simpler grammars, like Example 6.4 and Example 6.5, the definition is sufficient [11]. But for more complex grammars a better definition is needed.

Defining boolean grammars in terms of rewriting is not easy. It is easier to define these grammars in terms of language equations, as we show in the next section.

6.2 Boolean grammars in terms of Language Equations

Suppose we extend Definition 5.13 (Language equations for conjunctive grammars [13]) to include the complement function. Then we would say that for the system of language equations $\mathcal{E}_G$ there exists a least solution. However, this only holds if the function $F : \mathcal{P}(\Sigma^*)^{|N|} \to \mathcal{P}(\Sigma^*)^{|N|}$ is still Scott continuous.

Lemma 6.7 (*Complement operation is not Scott continuous*)

Let $F : \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Sigma^*)$ be the function with $F(\mathrm{A}) = \overline{\mathrm{A}}$ and $\Sigma = \{a\}$. Function F is not Scott continuous.

Proof:
We consider the chain $\mathrm{A}_1 \subseteq \mathrm{A}_2 \subseteq ... = \{a\} \subseteq \{a, aa\} \subseteq ...$
If the function is Scott continuous, it should hold that $F(\bigcup_i \mathrm{A}_i) = \bigcup_i F(\mathrm{A}_i)$.
However,
$F(\bigcup_i \mathrm{A}_i) = F(\Sigma^*) = \overline{\Sigma^*} = \emptyset$ and
$\bigcup_i F(\mathrm{A}_i) = F(\{a\}) \cup F(\{a, aa\}) \cup ... = \overline{\{a\}} \cup \overline{\{a, aa\}} \cup ....$
$= (\Sigma^* \setminus \{a\}) \cup (\Sigma^* \setminus \{a, aa\}) \cup ... = \Sigma^* \setminus \{a\}$
But $\emptyset \neq \Sigma^* \setminus \{a\}$, so the function is not Scott continuous.

This proves Lemma 6.7.

Lemma 6.7 shows that the complement function is not Scott continuous. This means that Kleene's Fixed Point Theorem generally does not apply to boolean grammars that use the complement operation, and that there is no least solution for these grammars.

This becomes a problem when there are systems with multiple solutions. The question rises what solution should be selected as the solution of the language equations. It does not make sense to allow multiple solutions, because it would become unclear what language the grammar is defining, which is the whole point of a formal grammar.

To resolve this problem, one approach is to define a boolean grammar as a grammar with a unique solution [11]. Grammars that have multiple solutions or none are not considered well-defined boolean grammars.

Example 6.8

Let $\mathcal{E}_{G_{11}}$ be a boolean grammar with $\Sigma = \{a\}$ and the following system of equations:

$$\begin{cases} \mathrm{S} = \overline{\mathrm{S}} \cap \{a\}\mathrm{A} \\ \mathrm{A} = \mathrm{A} \end{cases}$$

The unique solution for this system can be found with the function $F : \mathcal{P}(\Sigma^*)^2 \to \mathcal{P}(\Sigma^*)^2$ where $F(\mathrm{S}, \mathrm{A}) = (\mathrm{S}, \mathrm{A})$. The solution is $\mathrm{S} = \mathrm{A} = \emptyset$.

Since the language generated by S is defined by the complement of the language generated by S, the only solution for S is the empty set $\emptyset$. We get $\mathrm{S} = \emptyset = \overline{\emptyset} \cap \{a\}\mathrm{A} = \Sigma^* \cap \{a\}\mathrm{A}$. In order for the equation $\emptyset = \Sigma^* \cap \{a\}\mathrm{A}$ to hold, the only solution is that $\mathrm{A} = \emptyset$. We know from Definition 3.9 that concatenating with the empty set results in the empty set, and it is obvious that $\Sigma^* \cap \emptyset = \emptyset = \mathrm{S}$.

In conclusion, the system has a unique solution with $\mathrm{S} = \mathrm{A} = \emptyset$.

But something strange is happening in this example. Suppose we have a word $w \in \mathrm{A}$. We get the following equations:

$$\begin{cases} \mathrm{S} = \overline{\mathrm{S}} \cap \{a\}w \\ w = w \end{cases}$$

In order to have $aw \in \mathrm{S}$, it must be that $aw \in \overline{\mathrm{S}}$. This is a contradiction, so the conclusion must be that $w \notin \mathrm{A}$.

In order to reach this conclusion, we had to consider the word aw. But this word is longer than w. This differs significantly from the context-free grammars that we want to extend without losing important properties of these grammars. The context-free grammars define the words recursively. But in this case the string w does not depend on a shorter substring, but rather on a larger string.

This shows that it is not enough to restrict the definition of boolean grammars in terms of language equations to have a unique solution. In order to soundly define boolean grammars, Okhotin introduces a second restriction [10]. Instead of considering the strings of any length, we only consider the strings of length up to abd including ℓ (see Definition 3.4).

Definition 6.9 (*Well-formed boolean grammars [11]*)

Let $G = (\Sigma, N, R, S)$ be a boolean grammar. We consider the associated system of equations $\mathcal{E}_G$:

$$\mathrm{A} = \bigcup_{\substack{\mathrm{A} \to \alpha_1 \,\&\ldots\&\, \alpha_m \\ \&\, \neg\beta_1 \,\&\, \ldots \,\&\, \neg\beta_n \in R}} \left(\bigcap_{i=1}^{m} \alpha_i \cap \bigcap_{j=1}^{n} \overline{\beta_j} \right)$$

for all nonterminals $\mathrm{A} \in N$ as unknown languages, and for all $a \in \Sigma \cup \{\varepsilon\}, a$ denotes $\{a\}$.

For every $\ell \geq 0$ we consider the function over the nonterminals N:
$F_\ell : \mathcal{P}(\Sigma^{\leq \ell})^{|N|} \to \mathcal{P}(\Sigma^{\leq \ell})^{|N|}$, derived from the equations of $\mathcal{E}_G$.

In this thesis we call G a *well-formed boolean grammar* if F_ℓ has a unique solution (fixed point) for every ℓ. Then $\mathcal{E}_G$ is said to have a strongly unique solution, which is the union of the solutions for every ℓ.

A word $w \in \Sigma^*$ is in the language generated by the grammar when it is in the strongly unique solution (of the startsymbol S).

Example 6.10

Recall the boolean grammar $\mathcal{E}_{G_{11}}$, with $\Sigma = \{a\}$ and the following system of language equations:

$$\begin{cases} \mathrm{S} = \overline{\mathrm{S}} \cap \{a\}\mathrm{A} \\ \mathrm{A} = \mathrm{A} \end{cases}$$

The function $F : \mathcal{P}(\Sigma^{\leq \ell})^2 \to \mathcal{P}(\Sigma^{\leq \ell})^2$ has two solutions where $F(\mathrm{S}, \mathrm{A}) = (\mathrm{S}, \mathrm{A})$:

The first solution is the same as in Example 6.8: $\mathrm{S} = \mathrm{A} = \emptyset$
The second solution is $\mathrm{S} = \emptyset, \mathrm{A} = \{a^\ell\}$, because:

$$\begin{aligned} \mathrm{S} = \emptyset &= \emptyset \cap \{a\}\{a^\ell\} \\ &= \emptyset \cap \{a^{\ell+1}\} \\ &= \emptyset \cap \emptyset \qquad \textit{because } |a^{\ell+1}| > \ell \\ &= \emptyset \end{aligned}$$

$$\mathrm{A} = \{a^\ell\} = \{a^\ell\}$$

Since there are two solutions, the grammar is not well-formed according to Definition 6.9.

Example 6.11

The following boolean grammar $\mathcal{E}_{G_9}$ generates the language $\mathrm{L}_9 = \{ww \mid w \in \{a,b\}^*\}$, with $\Sigma = \{a,b\}$:

$$\begin{cases} \mathrm{S} = \overline{\mathrm{AB}} \cap \overline{\mathrm{BA}} \cap \mathrm{C} \\ \mathrm{A} = \mathrm{XAX} \cup \{a\} \\ \mathrm{B} = \mathrm{XBX} \cup \{b\} \\ \mathrm{C} = \mathrm{XXC} \cup \{\varepsilon\} \\ \mathrm{X} = \{a\} \cup \{b\} \end{cases}$$

The function $F : \mathcal{P}(\Sigma^{\leq \ell})^5 \to \mathcal{P}(\Sigma^{\leq \ell})^5$ has a unique solution where $F(\mathrm{S},\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{X}) = (\mathrm{S},\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{X})$, and it can be shown that it is:

$$\begin{aligned} \mathrm{S} &= \{ww \mid w \in \{a,b\}^*\} \\ \mathrm{A} &= \{xay \mid x,y \in \{a,b\}^*, |x| = |y|\} \\ \mathrm{B} &= \{xby \mid x,y \in \{a,b\}^*, |x| = |y|\} \\ \mathrm{C} &= \{xy \mid x,y \in \{a,b\}^*, |x| = |y|\} \\ \mathrm{X} &= \{w \mid w \in \{a,b\}\} \end{aligned}$$

When $\ell = 4$ we get the following:

- The nonterminal X generates the set $\{a,b\}$.
- The nonterminal C generates the set $\{aa, ab, bb, ba, aaaa, aaab, aabb, abbb, bbbb, bbba, bbaa, baaa\}$.
- The nonterminal B generates the set $\{b, aba, abb, bbb, bba\}$.
- The nonterminal A generates the set $\{a, aaa, aab, bab, baa\}$.
- The nonterminals AB generate the set $\{ab, aaba, aabb, abbb, abba\}$.
- The nonterminals BA generate the set $\{ba, baaa, baab, bbab, bbaa\}$.
- The resulting language generated by the startsymbol S is the set $\{aa, bb, aaaa, aabb, bbbb, bbaa\}$.

6.2.1 Limitations

Definition 6.9 provides a way to extend the conjunctive grammars. This allows for more languages to be defined. But this definition has several limitations as stated by Okhotin [11].

First of all, some grammars are deemed ill-formed according to the definition, while they obviously should be accepted to the definition. The easiest example of this fact is that the grammar with production rule $\mathrm{S} \to \mathrm{S}$ has many solutions. This result is not preferable, it would be better to define boolean grammars in such a way that there exists a least solution, rather than a unique one (as is the case with context-free grammars and conjunctive grammars).

Another great limitation is that it cannot be effectively decided whether a boolean grammar satisfies the conditions of being well-formed [10, Theorem 2].

Kountouriotis [7] has shown that the definition of Okhotin behaves strangely. Consider the following example:

Example 6.12 (*Kountouriotis [7]*)

Let $\mathcal{E}_{G_{12}}$ be a boolean grammar with $\Sigma = \{0, 1\}$, and the following system of language equations:

$$\begin{cases} \mathrm{S} = \overline{\mathrm{S}} \cap \overline{\mathrm{A}} \\ \mathrm{A} = \{0\} \cap \{1\} \end{cases}$$

This system of equations has no solution, and is therefore according to Definition 6.9 deemed ill-formed.

However, when we augment grammar $\mathcal{E}_{G_{12}}$ with the production rule $\mathrm{A} \to \mathrm{A}$, which intuitively does not add any value to the grammar, the system has a unique solution: $(\mathrm{S}, \mathrm{A}) = (\emptyset, \Sigma^*)$, and is therefore considered well-formed.

But if we instead augment the grammar with the production rule $\mathrm{S} \to \mathrm{S}$, which similarly does not add any value to the grammar, the system has a different unique solution: $(\mathrm{S}, \mathrm{A}) = (\Sigma^*, \emptyset)$, and also considered well-formed.

These three seemingly equivalent grammar definitions have three completely meanings according to Definition 6.9. The last two solutions are each other's complement. Obviously, this is not preferable behaviour.

Finally, Kountouriotis [7] shows that it is possible to define languages that are considered well-formed according to Definition 6.9, but generate strange languages. That is why he proposed a different definition of boolean grammars, which we consider in Chapter 7.

Chapter 7

Related Work

In this chapter we examine some related work to the conjunctive and boolean grammars.

7.1 Three-valued languages

In 2009, Kountouriotis [7] introduced an alternative for Okhotin's definition of boolean grammars (Definition 6.9). He proposed that, instead of basing the definition on two-valued languages, it would be better to base it on *three-valued languages*. Two-valued languages define a value for every word for a grammar: the word is either in the language generated by the grammar, or is not. The definitions of context-free grammars, conjunctive and boolean grammars as shown in this thesis, are based on this logic.

Kountouriotis proposes to consider instead three-valued languages. The three values are: a word is in the language (denoted by 1), a word is not in the language (denoted by 0), or a word is undefined in the language (denoted by $\frac{1}{2}$). Defining boolean grammars in terms of three-valued language equations has a very big advantage: the solution of the system can be found through a least fixed point.

A three-valued language L is denoted by a pair (L_1, L_2), with $L_1 \subseteq L_2 \subseteq \Sigma^*$, where L_1 represents the lower bound and L_2 the upper bound of the three valued language. Whenever $L_1 = L_2$, the language is said to be *completely defined*, and $L = (\emptyset, \Sigma^*)$ denotes a language where all the words are undefined. Determining the value of a word over such a three-valued language goes according to these rules, for any word w:

- $w \notin L_1 \wedge w \notin L_2$ implies that w is not a member of the three-valued language, we write $L(w) = 0$.
- $w \notin L_1 \wedge w \in L_2$ implies that it is undefined whether w is a member

of the three-valued language, we write $\mathrm{L}(w) = \frac{1}{2}$.

- $w \in \mathrm{L}_1$ implies that $w \in \mathrm{L}_2$, and therefore w is a member of the three-valued language, we write $\mathrm{L}(w) = 1$.

Kountouriotis defines two partial orderings on the three-valued languages, on which a pair of languages $\mathrm{L} = (\mathrm{L}_1, \mathrm{L}_2)$ and $\mathrm{K} = (\mathrm{K}_1, \mathrm{K}_2)$ can be compared.

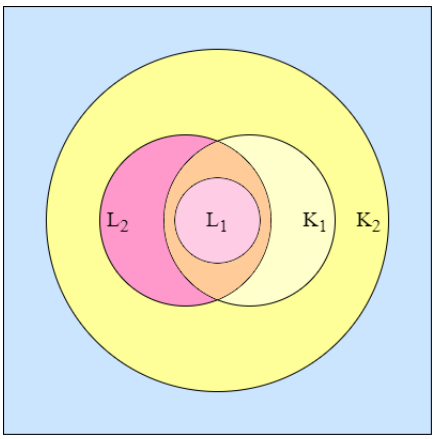


Figure 7.1: Venn diagram for the truth ordering

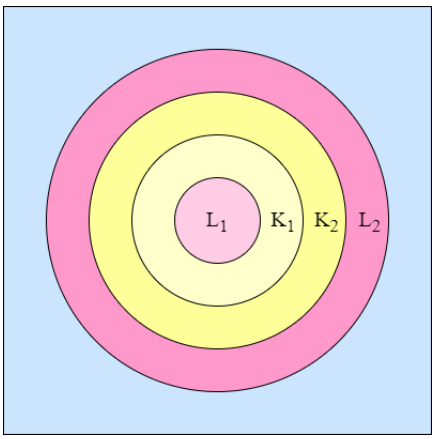


Figure 7.2: Venn diagram for the information ordering

The first ordering is based on the degree of truth: the *truth ordering*, denoted by $\mathrm{L} \sqsubseteq_T \mathrm{K}$, where $\mathrm{L}_1 \subseteq \mathrm{K}_1$ and $\mathrm{L}_2 \subseteq \mathrm{K}_2$. The relation $\mathrm{L} \sqsubseteq_T \mathrm{K}$ is defined as follows:

- $\mathrm{L}(w) = 0$ implies that $\mathrm{K}(w) = 0 \vee \mathrm{K}(w) = \frac{1}{2} \vee \mathrm{K}(w) = 1$.
- $\mathrm{L}(w) = \frac{1}{2}$ implies that $\mathrm{K}(w) = \frac{1}{2} \vee \mathrm{K}(w) = 1$.
- $\mathrm{L}(w) = 1$ implies that $\mathrm{K}(w) = 1$.

Figure 7.1 shows the Venn diagram for the two languages in this partial ordering. The least element of the truth ordering of the language equations is $(\emptyset, \emptyset)^{|N|}$, over the nonterminals N of the grammar: every language is completely defined as $\emptyset$. The greatest element is $(\Sigma^*, \Sigma^*)^{|N|}$. The operations concatenation, union and intersection are Scott continuous with respect to the truth ordering. Complementation is not monotone.

The second ordering is with respect to the degree of information: the *information ordering*, denoted by $\mathrm{L} \sqsubseteq_I \mathrm{K}$, where $\mathrm{L}_1 \subseteq \mathrm{K}_1$ and $\mathrm{K}_2 \subseteq \mathrm{L}_2$. The relation $\mathrm{L} \sqsubseteq_I \mathrm{K}$ is defined as follows:

- $\mathrm{L}(w) = 0$ implies that $\mathrm{K}(w) = 0$.
- $\mathrm{L}(w) = \frac{1}{2}$ implies that $\mathrm{K}(w) = 0 \vee \mathrm{K}(w) = \frac{1}{2} \vee \mathrm{K}(w) = 1$.

- $\mathrm{L}(w) = 1$ implies that $\mathrm{K}(w) = 1$.

Figure 7.2 shows the Venn diagram for the two languages in this partial ordering. The least element of the information ordering of the language equations is $(\emptyset, \Sigma^*)^{|N|}$, over the nonterminals N of the grammar: every language is completely undefined. The operations concatenation, union, intersection and complementation are Scott continuous with respect to the information ordering.

Kountouriotis uses both the orderings to define boolean grammars in terms of language equations in such a way that there is a two-level fixed point, and therefore a solution to the system.

7.2 The Chomsky hierarchy

Chomsky [2] has defined a famous hierarchy of classes of formal languages in 1959. Table 7.1 shows this hierarchy. It should be noted that Type-3 $\subseteq$ Type-2 $\subseteq$ Type-1 $\subseteq$ Type-0.

Grammar Type	Language
Type-0	Recursively enumerable
Type-1	Context-sensitive
Type-2	Context-free
Type-3	Regular

Table 7.1: The Chomsky hierarchy

Context-sensitive grammars have proven to be hard to parse [3], and therefore hardly ever used in practice. Okhotin [10] proposes therefore to ommit this hierarchy, and classify grammars differently, based on their computational complexity classes of the *membership problem*: the complexity of calculating whether some word is a member of the language that the grammar describes. For context-sensitive grammars the computational complexity class is in **PSPACE** [6]. That of conjunctive and boolean grammars is in **P** [10].

His proposal is shown in Figure 7.3, for the language specification formalisms: finite automata (Reg), linear context-free grammars (LinCF), context-free grammars (CF), linear conjunctive grammars (LinConj), conjunctive grammars (Conj), boolean grammars (Bool), deterministic context-sensitive grammars (DetCS) and context-sensitive grammars (CS). Note that Conj languages $\subseteq$ Bool languages $\subseteq$ DetCS languages $\subseteq$ CS languages.

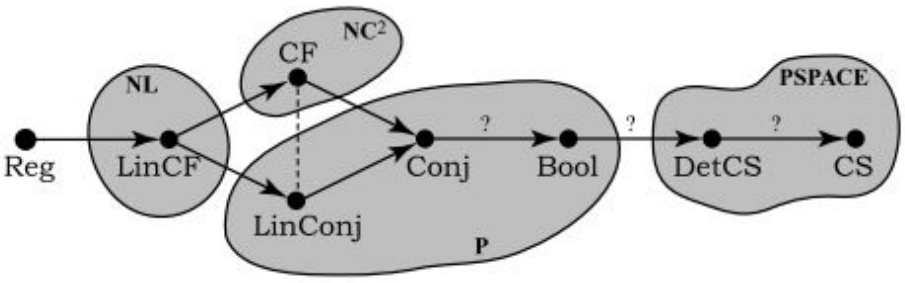


Figure 7.3: The proposed hierarchy of Okhotin [10]

Chapter 8

Conclusions

We have shown that the extensions to context-free grammars offer possibilities to define more languages than just context-free languages, while still maintaining the main principle of context-free grammars: the production rules can always be applied, regardless of the context in which they occur.

The definition of conjunctive grammars allows us to freely use the conjunction operation, similar to how disjunction can be used in context-free grammars. The result is that many languages that are not context-free can be defined using the conjunctive grammars, while remaining as intuitive to use as context-free grammars are.

We have seen multiple examples of conjunctive grammars, the easier ones define the intersection of two context-free grammars. For a more complex conjunctive grammar, we have proven what language it generates. Furthermore, we have concluded that parsing conjunctive grammars has the same upper bounds as context-free grammars.

We have shown that extending conjunctive grammars with the negation operation is not as simple as extending the context-free grammars with the conjunction operation. The definition of well-formed boolean grammars as defined by Okhotin [10] is not perfect. However, it illustrates that using the negation operation in grammars provides a method to define grammars for complex languages.

These relatively new definitions are still being refined, but show great promise. They encourage to further investigate how to define formal grammars, and challenge the standard convention of formal grammars.

8.1 Future work

As this field of study is relatively new, refinements can still be made. In 2006 Okhotin presented 9 theoretical open problems for conjunctive and boolean grammars[1]. Since then, two of those problems have been solved. He offers a reward for solving each of the remaining problems.

1. Are there any languages recognized by deterministic linear bounded automata working in time $O(n^2)$ that cannot be specified by Boolean grammars?

2. Do conjunctive grammars over a one-letter alphabet generate only regular languages? **(solved negatively by Jeż in 2007)** [5]

3. Are the languages generated by Boolean grammars contained in $\mathsf{DTIME}(n^{3-\varepsilon})$ for any $\varepsilon > 0$? **(solved positively by Okhotin in 2009)** [9]

4. Are the languages generated by Boolean grammars contained in $\mathsf{DSPACE}(n^{1-\varepsilon})$ for any $\varepsilon > 0$?

5. Is it true that for every Boolean grammar there exists a Boolean grammar in Greibach normal form that generates the same language?

6. Is the family of conjunctive languages closed under complementation?

7. Do there exist any inherently ambiguous languages with respect to Boolean grammars?

8. Does there exist a number $k_0 > 0$, such that, for all $k \geq k_0$, Boolean LL(k) grammars generate the same family of languages as Boolean LL(k_0) grammars?

9. Does there exist a number $k \geq 0$, such that every language generated by any Boolean grammar can be generated by a k-nonterminal Boolean grammar?

[1] https://users.utu.fi/aleokh/boolean/nine_open_problems.html

Bibliography

[1] Jean-Michel Autebert, Jean Berstel, and Luc Boasson. "Context-Free Languages and Pushdown Automata". In: *Handbook of Formal Languages: Volume 1 Word, Language, Grammar*. Ed. by Grzegorz Rozenberg and Arto Salomaa. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 111–174. ISBN: 978-3-642-59136-5. DOI: 10.1007/978-3-642-59136-5_3. URL: https://doi.org/10.1007/978-3-642-59136-5_3.

[2] Noam Chomsky. "On certain formal properties of grammars". In: *Information and Control* 2.2 (1959), pp. 137–167. ISSN: 0019-9958. DOI: https://doi.org/10.1016/S0019-9958(59)90362-6. URL: https://www.sciencedirect.com/science/article/pii/S0019995859903626.

[3] Elias Dahlhaus and Manfred K. Warmuth. "Membership for growing context-sensitive grammars is polynomial". In: *Journal of Computer and System Sciences* 33.3 (1986), pp. 456–472. ISSN: 0022-0000. DOI: https://doi.org/10.1016/0022-0000(86)90062-0. URL: https://www.sciencedirect.com/science/article/pii/0022000086900620.

[4] Seymour Ginsburg and H Gordon Rice. "Two families of languages related to ALGOL". In: *Journal of the ACM (JACM)* 9.3 (1962), pp. 350–371.

[5] Artur Jeż. "Conjunctive grammars generate non-regular unary languages". In: *International Journal of Foundations of Computer Science* 19.03 (2008), pp. 597–615.

[6] Richard M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*. Ed. by Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger. Boston, MA: Springer US, 1972, pp. 85–103. ISBN: 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2_9. URL: https://doi.org/10.1007/978-1-4684-2001-2_9.

[7] Vassilis Kountouriotis, Christos Nomikos, and Panos Rondogiannis. "Well-founded semantics for Boolean grammars". In: *Information and Computation* 207.9 (2009), pp. 945–967. ISSN: 0890-5401. DOI: https://doi.org/10.1016/j.ic.2009.05.002. URL: https://www.sciencedirect.com/science/article/pii/S0890540109001473.

[8] H.R. Nielson and F. Nielson. *Semantics with Applications: An Appetizer*. Undergraduate Topics in Computer Science. Springer London, 2007. ISBN: 9781846286919. URL: https://books.google.nl/books?id=3dudDAEACAAJ.

[9] Alexander Okhotin. "A Tale of Conjunctive Grammars". In: *Developments in Language Theory*. Ed. by Mizuho Hoshi and Shinnosuke Seki. Cham: Springer International Publishing, 2018, pp. 36–59. ISBN: 978-3-319-98654-8.

[10] Alexander Okhotin. "Boolean grammars". In: *Information and Computation* 194.1 (2004), pp. 19–48. ISSN: 0890-5401. DOI: https://doi.org/10.1016/j.ic.2004.03.006. URL: https://www.sciencedirect.com/science/article/pii/S0890540104001075.

[11] Alexander Okhotin. "Conjunctive and Boolean grammars: The true general case of the context-free grammars". In: *Computer Science Review* 9 (2013), pp. 27–59. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2013.06.001. URL: https://www.sciencedirect.com/science/article/pii/S157401371300018X.

[12] Alexander Okhotin. "Conjunctive grammars". In: *Journal of Automata, Languages and Combinatorics* 6.4 (2001), pp. 519–535.

[13] Alexander Okhotin. "Conjunctive grammars and systems of language equations". In: *Programming and Computer Software* 28.5 (2002), pp. 243–249.

[14] Alexander Okhotin. "The dual of concatenation". In: *Theoretical Computer Science* 345.2 (2005). Mathematical Foundations of Computer Science 2004, pp. 425–447. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2005.07.019. URL: https://www.sciencedirect.com/science/article/pii/S0304397505004056.

[15] K.H. Rosen and K. Krithivasan. *Discrete Mathematics and Its Applications*. McGraw-Hill, 2013. ISBN: 9780071315012. URL: https://books.google.nl/books?id=ZO8iMAEACAAJ.

[16] Stephen Scheinberg. "Note on the boolean properties of context free languages". In: *Information and Control* 3.4 (1960), pp. 372–375. ISSN: 0019-9958. DOI: https://doi.org/10.1016/S0019-9958(60)90965-7. URL: https://www.sciencedirect.com/science/article/pii/S0019995860909657.

[17] Dana Scott. “Continuous lattices”. In: *Toposes, Algebraic Geometry and Logic*. Ed. by F. W. Lawvere. Berlin, Heidelberg: Springer Berlin Heidelberg, 1972, pp. 97–136. ISBN: 978-3-540-37609-5.

[18] Stuart M. Shieber, Yves Schabes, and Fernando C.N. Pereira. “Principles and implementation of deductive parsing”. In: *The Journal of Logic Programming* 24.1 (1995). Computational Linguistics and Logic Programming, pp. 3–36. ISSN: 0743-1066. DOI: https://doi.org/10.1016/0743-1066(95)00035-I. URL: https://www.sciencedirect.com/science/article/pii/074310669500035I.

[19] Thomas A. Sudkamp. *Languages and Machines: An Introduction to the Theory of Computer Science (3rd Edition)*. USA: Addison-Wesley Longman Publishing Co., Inc., 2005, pp. 41–102. ISBN: 0321322215.

[20] Detlef Wotschke. “The Boolean Closures of the Deterministic and Nondeterministic Context-Free Languages”. In: *GI Gesellschaft für Informatik e. V.: 3. Jahrestagung Hamburg, 8.–10. Oktober 1973*. Ed. by Wilfried Brauer. Berlin, Heidelberg: Springer Berlin Heidelberg, 1973, pp. 113–121. ISBN: 978-3-662-41148-3. DOI: 10.1007/978-3-662-41148-3_11. URL: https://doi.org/10.1007/978-3-662-41148-3_11.

List of Theorems