Classifying sequences by combining context-free grammars and OWL ontologies

Nicolas Lazzari[1,3][0000−0002−1601−7689], Andrea Poltronieri[2][0000−0003−3848−7574], and Valentina Presutti[1⋆][0000−0002−9380−5160]

[1] LILEC, University of Bologna, Italy
nicolas.lazzari2@studio.unibo.it, valentina.presutti@unibo.it
[2] Department of Computer Science and Engineering, University of Bologna, Italy
andrea.poltronieri2@unibo.it
[3] Department of Computer Science, University of Pisa, Italy

Abstract. This paper describes a pattern to formalise context-free grammars in OWL and its use for sequence classification. The proposed approach is compared to existing methods in terms of computational complexity as well as pragmatic applicability, with examples in the music domain.

Keywords: sequence classification · context-free grammar · ontologies · music

1 Introduction

The introduction of formal grammars by Chomsky in the 50s [7], and in particular Context Free Grammars (CFG), led to prolific research in the area of Natural Language Processing. Methods based on statistical language modeling have mostly replaced formal grammars, nevertheless research in this area is still relevant as many domains and tasks benefit from their application. For example, an important application of formal grammars concerns high level programming languages. Through an efficient parsing process [1], machine-level instructions can be abstracted in human readable instructions. Theoretically, every problem that can be abstracted as a *sequence of symbols* can be modeled with formal grammars, which makes them a suitable tool for *sequence classification*, the task we focus on in this paper. In the biology field, the classification of RNA secondary structures has been performed using CFG [9, 15, 35]. Similarly, in the music field, CFG are used to classify different types of harmonic and melodic sequences [3, 24, 36, 39, 41, 42].

Background. A *language* is a collection of sequences, each defined according to a finite set of symbols [8]. A *grammar* can be interpreted as a function of a language, having a set of symbols as its domain, and a set of sequences as its range. Defining a formal grammar is a complex task that requires deep knowledge of the application domain as well as good modeling skills, to obtain

⋆ Alphabetical order

an efficiently parsable grammar. CFGs can be parsed in less than $O(n^3)$ [43], with n the length of the string. However when symbols have ambiguous semantics and require additional attributes to be disambiguated, parsing a sequence is NP-complete [14]. Ambiguous symbols are a common issue, for instance in the case of polysemous words in natural language or diminished chords in music. This problem can be mitigated through the use of complex notations, such as SMILES [44] to represent molecules in the biology field or Harte [18] to represent musical chords. Nevertheless, these notations are either hard to interpret, requiring additional tools to be converted back into a human understandable format, or they cause loss of information.

We address the problem of sequence classification by proposing a hybrid approach that combines the use of Context Free Grammar (CFG) parsers with OWL ontologies. We define a pattern to formalise CFG by providing a novel definition for CFG based on Description Logic. We define a set of algorithms to produce an OWL ontology based on this pattern that supports the alignment of symbols in a sequence to its classes. Our approach is based on the identification of sub-sequences according to the taxonomy defined in the ontology. We argue that our proposal has a relevant pragmatic potential as it enables sequence classification based on semantic web knowledge representation, therefore supporting the linking of Context Free Grammars to web ontologies and knowledge graphs.

The contribution of this research can be summarised as follows:

- defining a novel formalisation of Context Free Grammars based on Description Logic;
- providing an algorithm for the conversion of such formalisation in OWL;
- demonstrating the correctness, computational complexity and applicability of the proposed method in the music domain.

The paper is organized as follows: in Section 2, an overview of related works on sequence classification is presented. Section 3 provides relevant definitions for Context Free Grammars that are used later in Section 4, to describe the formalisation of CFGs in DL. Section 5 describes our approach to sequence classification. In section 6 we evaluate our method on the task of sequence classification in the music domain. Finally, in Section 7, we summarize the contribution and discuss future development.

2 Related Work

Relevant work to this contribution include: techniques for sequence classification, sequence model with grammars, approaches to integrate CFG and semantic web technologies, and their application in the music domain (cf. Section 6).

Sequence Classification (SQ) is the task of predicting the class of an input, defined as a sequence over time or space, among a predefined set of classes [31]. SQ is relevant in several application fields, such as genomics research [26], health

informatics, abnormality detection and information retrieval [47], Natural Language Processing (NLP) [6, 28]. In [47] different methodologies for SQ are identified, such as feature-based classification, sequence distance-based classification and support vector machines (SVM). The most advanced approaches mainly rely on Deep Learning (DL) [5, 30]. SQ is relevant in the music domain, as sequences are at its core, for instance melodic and harmonic sequences that span over a temporal dimension. An example of sequence classification applied to music is [31], which addresses the recognition of *raga* using recurrent neural networks (LSTM-RNN).

Sequence Model with Grammars. There is a close relationship between sequences and formal grammars. A classic related task (ranging from natural language processing [27] to bio-informatics [9]) is *grammar inference* [49]. Grammars are used for the classification of sequences of different types, mainly for analysing genetic sequences [15, 35]. There are applications of grammars for music classification. The most renowned example is the generative theory of tonal music (GTTM) [29], analogous to Chomsky's transformational or generative grammar. [7]. Although GTTM does not explicitly provide generative grammar rules, this work has inspired the formalisation of a wide range of context-free rules, describing different music genres [24, 39, 41], melody [3] and harmony [36, 42]. These works are relevant input to our work as they formalise aspects of certain types of sequences into rules.

Sequence Model with OWL. There are proposals to use OWL to classify sequences. However, one of the main challenges when dealing with sequences in OWL is to organise the elements being described in an ordered fashion, as proposed in [12]. For instance, OWL reasoning is employed for classifying genomic data [46]. A method for analysing jazz chord sequences is proposed in [33]. This system is based on an ontology which, through reasoning, produces a hierarchical jazz sequence analysis. Similarly, two OWL ontologies, $\mathcal{MEO}$ and $\mathcal{SEQ}$, are presented in [45] that combined with a CFG parser support sequence classification. Nevertheless, this method is only able to represent *safely-concatenable* CFG, while we overcome this limitation in our approach (cf. Section 5.3).

3 Preliminaries

This section introduces the notation and the definitions used in Section 4, based on [20] that the reader can consult for details.

Definition 1 (Context Free Grammar). *A Context Free Grammar (CFG)* $G = (V, \Sigma, R, S)$ *consists of a finite set of non-terminal* V *(variables), a set of terminals* Σ *such that* $\Sigma \cap V = \emptyset$*, a set of functions* $R \subseteq V \times (V \times R)^*$ *(production), and a starting symbol* $S \in V$*.*

Definition 2 (Language of a grammar). *The language of a grammar* $G(V, \Sigma, R, S)$ *is defined as* $L(G) = \{w \in \Sigma^* : S \overset{*}{\Rightarrow} w\}$*, where* $S \overset{*}{\Rightarrow} w$ *represents the consec-*

utive application of production $f \in R$ starting from the initial symbol S, called derivation.

Example 1 (Context Free Grammar). Let $G = (V, \Sigma, R, S)$ with

$$
\begin{aligned}
V &= \{\text{Expression}, \text{Bit}\} \\
\Sigma &= \{0, 1, +\} \\
R &= \{\text{Expression} \to \text{Expression} + \text{Expression} \mid \text{Bit } 0 \mid \text{Bit } 1 \mid 0 \mid 1, \\
&\qquad \text{Bit} \to \text{Bit } 0 \mid \text{Bit } 1 \mid 0 \mid 1 \ \} \\
S &= \ \text{Expression}
\end{aligned}
$$

where $X \to X_1 | \cdots | X_n$ is a shorthand for $\{X \to X_1, \ \cdots \ X \to X_n\}$

Example 1 shows a simple grammar used to parse the sum of two binary numbers. Its language L, as defined in definition 2, is of the form $L = \{0+0, 0+1, 1+0, 10+0, \cdots, 11010+10, \cdots\}$.

In order to express a concise and effective conversion method and its corresponding proof we only consider grammars in Chomsky Normal Form, as defined in Definition 3. This results in homogeneous productions in the form

$$A \to BC \quad \text{or} \quad A \to t$$

where $A, B, C \in V$ and $t \in \Sigma$.

Definition 3 (Chomsky Normal Form). *A Context Free Grammar is in Chomsky Normal Form (CNF) if the set of functions $R \subseteq V \times (V \backslash \{S\} \times R)^2$*

Note that the imposed restriction does not imply any loss in expressiveness, since any context-free grammar can be converted in CNF [20]. For instance, Example 1, converted in CNF, results in the grammar in Example 2.

Example 2 (Example 1 in Chomsky Normal Form). Let $G = (V, \Sigma, R, S)$ with

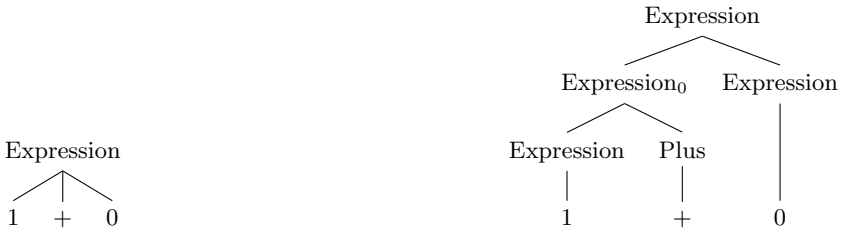
$$
\begin{aligned}
V &= \{\text{Expression}, \text{Expression}_0, \text{Bit}, \text{Zero}, \text{One}, \text{Plus}\} \\
\Sigma &= \{0, 1, +\} \\
R &= \{\text{Expression} \to \text{Expression}_0 \ \text{Expression} \mid \text{Bit Zero} \mid \text{Bit One} \mid 0 \mid 1, \\
&\qquad \text{Expression}_0 \to \text{Expression Plus}, \\
&\qquad \text{Bit} \to \text{Bit Zero} \mid \text{Bit One} \mid 0 \mid 1, \\
&\qquad \text{Plus} \to +, \\
&\qquad \text{Zero} \to 0, \\
&\qquad \text{One} \to 1\}
\end{aligned}
$$

When parsing a language based on a grammar, it is useful to visualise the derivation process as a parse tree.

Definition 4 (Parse tree of a CFG). *A parse tree T of $G = (V, \Sigma, R, S)$ is a tree in which each leaf $l \in (V \cup \Sigma)$ and each inner node $n_i \in V$. Given $c_1 \cdots c_n$ the children of an inner node n_i then $\exists f \in R$ s.t. $f : n_i \to c_1 \cdots c_n$. [20]*

Corollary 1. *Given T the parse tree of a CFG in CNF $\Rightarrow$ T is a binary tree.*

Corollary 1 follows from Definitions 3 and 4, since each production is either a unary or a binary function. Figure 1a shows the parse tree of the sequence *1+0* from the grammar defined by Example 1 and Figure 1b shows the parse tree the grammar defined by Example 2.



(a) Parse tree generated by grammar from Example 2

(b) Parse tree generated by grammar from Example 2

Fig. 1: Parse trees obtained from the sequence *1+0*

4 Formalising Context-Free Grammars using Description Logic

As OWL is based on Description Logic (DL) theory, we define a DL-based formalisation of Context-Free Grammars (in Chomsky Normal Form), which we refer to as CFG-DL. CFG-DL is based on Definition 1, where variables and terminals are represented as concepts[4]. We demonstrate that any CFG can be converted in a CFG-DL (cf. Theorem 1) and that such conversion can be performed in $O(n)$ (cf. Theorem 2). Theorem 1 and its proof rely on the concept of *rolification*, which formalises axioms that act as rules in the form *if-then* [25]. For each concept C a corresponding axiom R_C is created and the restriction $C \equiv R_C.Self$ is imposed. By chaining together different axioms it is possible to define *if-then* rules. For a more in-depth explanation, please refer to Krisnadhi et al. [25].

Definition 5 (CFG-DL). *A CFG-DL $G_{DL} = (C_v, R, C_\Sigma, S)$ consists of a finite set of concepts C_v, a finite set of concepts C_Σ, a set of axioms R, and a starting concept $S \in C_v$.*

Theorem 1. *Every Context-Free Grammar G in Chomsky Normal Form can be converted in a CFG-DL G_{DL}.*

[4] DL concepts translate into OWL classes

Proof. Given a Context-Free grammar $G = (V, \Sigma, R, S)$ in Chomsky Normal Form we can obtain the corresponding $G_{DL} = (C'_v, R', C'_\Sigma, S')$ as follows:

1. $\forall v \in V$ let C_v be a concept such that $C_v \sqsubseteq C'_v$
   $\implies \forall v \in V \ \exists C_v \sqsubseteq C'_v$, where C_v is the respective concept of the variable V.
2. $\forall t \in \Sigma$ let C_t be a concept such that $C_t \sqsubseteq C'_\Sigma$
   $\implies \forall t \in \Sigma \ \exists C_t \sqsubseteq C'_\Sigma$, where C_t is the respective concept of the terminal t.
3. Let $f \in R$. It follows from Definition 3 that f is of either type:
   (a) $R \to AB$ such that $R \in V$, $A, B \in V \cup \Sigma$.
   (b) $R \to t$ such that $R \in V$ and $t \in \Sigma$;
   Both cases can respectively be represented in DL as follows:
   (a) i. Let $C_R \sqsubseteq C'_v, C_A \sqsubseteq C'_v, C_B \sqsubseteq C'_v$ be the respective concepts of R, A, B definied in step 1
   ii. Let R_R, R_A, R_B be the rolification [25] of the concepts C_R, C_A, C_B such that $C_R \equiv \exists R_R.Self$, $C_A \equiv \exists R_A.Self$, and $C_B \equiv \exists R_B.Self$.
   iii. Let R_{next} be the role such that $C_1 \circ R_{next} \circ C_2$ has semantic meaning C_1 *has as next element in the sequence* C_2, with $C_1 \sqsubseteq C'$ and $C_2 \sqsubseteq C'$
   iv. Let $V_1 \equiv \exists R_1.Self$ and $V_2 \equiv \exists R_2.Self$ be roles such that $R_A \circ R_{next} \circ R_B \sqsubseteq R_1$ and $R_B \circ R_{next}^{-1} \circ R_A \sqsubseteq R_2$.

   $\xrightarrow{3(a)i,3(a)ii,3(a)iii,3(a)iv} \quad R \to AB \iff (C_A \sqcap V_1) \sqcup (C_B \sqcap V_2) \sqsubseteq C_R.$

   (b) Let $C_t \sqsubseteq C'_\Sigma$ be the concept of the terminal t defined in step 2 and $C_R \sqsubseteq C'_V$ be the concept of variable R defined in step 1
   $\Rightarrow R \to t \iff C_t \sqsubseteq C_r$
   $\xrightarrow{3a,3b} \quad f \in R', \ \forall f \in R$.
4. $\xRightarrow{1}$ $\exists C_s \sqsubseteq C'_V$ where C_s is the concept corresponding to S, as defined in step 1.

$\xrightarrow{1,2,3,4} G_{DL} \equiv G.$ ■

Theorem 2. *The conversion between a CFG* $G = (V, \Sigma, R, S)$ *and a CFG-DL* $G_{DL} = (C'_v, R', C'_\Sigma, S')$ *can be performed in* $O(n)$*, in particular* $O(|V| + |\Sigma| + |R|)$*.*

Proof. It follows from the proof of Theorem 1 as we only need to loop through each element of V, Σ and R at most one time. ■

We remark that in Definition 5 terminals are modeled as concepts and stand at the same level of variables. At first, it might seem more intuitive to represent terminals as individuals. But this would radically change the semantic meaning of an element in a sequence. Take for example the sequence *10+11* from the language of grammar in Example 1. There are three occurrences of terminal *1*, but they are fundamentally different entities: the first occurrence of terminal

1 is characterized by its syntactic aspect as well as its position with respect to the whole sequence. If we represent each terminal as an individual then each occurrence of that terminal in a sequence would be represented by the very same individual. This would invalidate the semantics of the whole sequence and yield a wrong formalization. In order to address this issue, we need a proper definition of how to represent a sequence in description logic. We do that by adapting Definition 2 to CFG-DL.

Definition 6 (Language of a CFG-DL). *Let $G = (C_v, R, C_\Sigma, S)$ be a CFG-DL, we define as $L(G)$ the set of sequences $\overline{s}$ such that, given N the number of elements in the sequence $\overline{s}$, $\overline{s} \equiv (C_1 \sqcap \exists R_{next}.C_2) \sqcap \cdots \sqcap (C_{N-1} \sqcap \exists R_{next}.C_N)$, with R_{next} the role defined in step 3(a)iii of Theorem 1's proof and $C_t \sqsubseteq C_\Sigma, t \in [1, N]$.*

5 Sequence classification using CFG-DL

CFG-DL can be represented in OWL, as OWL2 direct semantics is based on Description Logic [22]. We devise an algorithm based on Definition 5 and on the respective constructive proof 4 of Theorem 1. Algorithm 1 converts a CFG to OWL, without generating any intermediary CFG-DL. A similar algorithm can be defined to convert a CFG in CFG-DL, following the constructive proof 4.

Triples are written in Manchester syntax [21]. We use the symbol ▶ to indicate the OWL triples that need to be created. We generally use $\overline{R}_{next}$ as the role R_{next} defined in step 3(a)iii of Thereom 1's proof. Any arbitrary OWL property can be used as $\overline{R}_{next}$ as long as it is a functional property, such as the *seq:directlyPrecedes* property from the *sequence* Ontology Design Pattern [19]. Algorithm 1 has also complexity $O(n)$: similarly to the considerations on Theorem 2, we only need to loop through each element of V, Σ and R at most one time.

The rolification of the classes $\overline{V}_1$ and $\overline{V}_2$ is performed by using the existential restriction on `owl:Thing`. This prevents the creation of non-simple properties due to the use of property chain later in the algorithm and allows the usage of reasoners such as Hermit [16] or Pellet [37].

Sequences must be converted to be used in the ontology obtained with Algorithm 1. Algorithm 2 presents an algorithm that performs such conversion in $O(n)$. It is based on Definition 6. Analogously to Algorithm 1, we express triples in Manchester syntax using the symbol ▶ and we use $\overline{R}_{next}$ as the role R_{next} defined in step 3(a)iii of Thereom 1's proof.

Figure 2, shows the grammar from Example 2, converted to a CFG-DL in OWL with Algorithm 1, used to parse the sequence *1 + 0*, converted using Algorithm 2. We can see how the whole sequence is correctly classified to be of class `Expression` and how `Expression_0` and `Expression` are classified as subclass of each other. Indeed, `Expression_0` and `Expression` are equivalent. This can be observed from the normalization process performed on Example 1 that resulted in Example 2: *Expression_0* variable is introduced to obtain

Algorithm 1 CFG in OWL

Require: $G = (V, \Sigma, R, S)$
- ▶ `ObjectProperty:` $\overline{R}_1$
- ▶ `ObjectProperty:` $\overline{R}_2$
- ▶ `Class:` $\overline{V}_1$ `EquivalentTo:` $\overline{R}_1$ `some`
- ▶ `Class:` $\overline{V}_2$ `EquivalentTo:` $\overline{R}_2$ `some`

for $\overline{v} \in V$ **do**
- ▶ `ObjectProperty:` $R_{\overline{v}}$
- ▶ `Class:` $C_{\overline{v}}$ `EquivalentTo:` $R_{\overline{v}}$ `some Self`

end for

for $\overline{t} \in \Sigma$ **do**
- ▶ `ObjectProperty:` $R_{\overline{t}}$
- ▶ `Class:` $C_{\overline{t}}$ `EquivalentTo:` $R_{\overline{t}}$ `some Self`

end for

for $r \in R$ **do**

if r is of type $R \to AB$ **then**

with $\overline{C}_R$, $\overline{C}_A$, $\overline{C}_B$ *being the respective concepts of* R, A, B

with $\overline{R}_A$, $\overline{R}_B$ *being the respective rolification of* A, B

- ▶ `ObjectProperty:` $\overline{R}_1$ `SubPropertyChain:` $\overline{R}_A$ `o` $\overline{R}_{next}$ `o` $\overline{R}_B$
- ▶ `ObjectProperty:` $\overline{R}_2$ `SubPropertyChain:` $\overline{R}_B$ `o inverse(`$\overline{R}_{next}$`)` `o` $\overline{R}_A$
- ▶ `(`$\overline{C}_A$ `and` $\overline{V}_1$`) or (`$\overline{C}_B$ `and` $\overline{V}_2$`) SubClassOf:` $\overline{C}_R$

else if r is of type $R \to t$ **then**

with $\overline{C}_R$, $\overline{C}_t$ *being the respective concepts of* R, t

- ▶ `Class:` $\overline{C}_t$ `SubClassOf:` $\overline{C}_R$

end if

end for

Algorithm 2 Sequence in OWL for CFG-DL

Require: $G = (V, \Sigma, R, S)$
Require: $s \subseteq \Sigma^*$ the sequence to represent
Require: N the length of the sequence s

for $i \in [1, N-1]$ **do**

$s_i \leftarrow s[i]$

$s_n \leftarrow s[i+1]$

with $\overline{C}_i$, $\overline{C}_n$ *being the respective concepts of the terminals* s_i, s_n

- ▶ `Individual:` s_n `Types:` $\overline{C}_n$
- ▶ `Individual:` s_i `Types:` $\overline{C}_i$ `Facts:` $\overline{R}_{next}$ s_n

end for

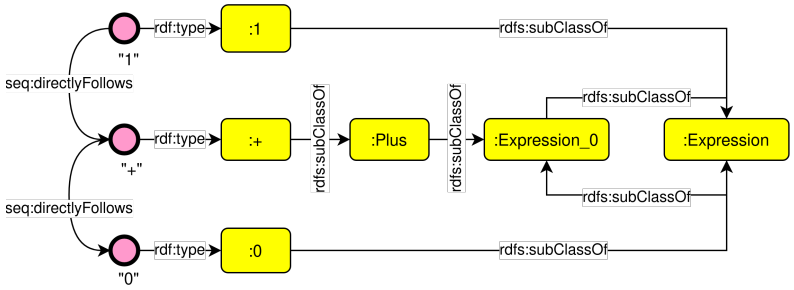
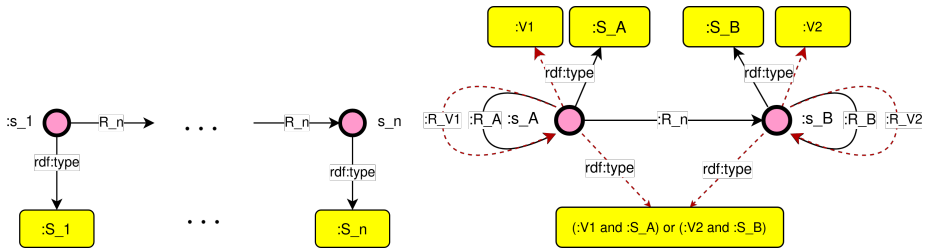


Fig. 2: Sequence *1+0* parsed by the grammar from example 2 represented as CFG-DL in OWL

a binary projection of $Expression \rightarrow Expression + Expression$, as required by CNF. If we substitute every occurrence of *Expression_0* with its right hand side ($Expression + Expression$) an equivalent grammar, which is not in CNF, is obtained. The overall pattern in Figure 3 can be generalized to every CFG converted in OWL.



(a) (b)

Fig. 3: General ontology patterns from Algorithms 1 (3a) and 2 (3b), for a rule of type $R \rightarrow A\ B$. The red arrows are created by the reasoner, due to the definition of the property chain and the general axiom. The pattern from a rule of type $R \rightarrow t$ is a simple subsumption relation.

5.1 Computational complexity

An ontology produced by Algorithm 1 is in DL $\mathcal{SROIEL}$, which is contained in OWL 2 DL [25]. It has exponential complexity (`NExpTime`) for automated reasoning [23]. *Parsing* a sequence through a DL reasoner would be too complex compared to an external parser: parsing CFGs has complexity $O(n^3)$ [43] even in the case of ambiguous grammars [13]. Without the use of inverse properties the produced ontology is in DL $\mathcal{SROEL}$, which is within OWL-EL and solvable

in polynomial time [23]. To mitigate the overall complexity we propose a hybrid approach combining CFG parsers and OWL reasoning, to perform sequence classification.

5.2 Combining CFG parser with OWL-based reasoning

We claim that converting CFG in CFG-DL, besides being an interesting theoretical approach, constitutes a relevant pragmatic approach to perform automatic sequence classification that can benefit from an explicit knowledge representation, using OWL ontologies. In practice, given a CFG G, after recognising a sequence $s \in L(G)$ using a parser for G, the resulting parse tree can be converted, using Algorithm 3 to instantiate the OWL ontology O resulting from Algorithm 1, for G.

Algorithm 3 Parse tree in OWL

Require: $G = (V, \Sigma, R, S)$ in CNF
Require: $s \subseteq \Sigma^*$ with $s \in L(G)$
Require: T the parse tree obtained by parsing s with G
Ensure: T is a binary tree
  for all leaf l in R **do**
    with $\overline{C}_l$ *being the concept of the terminal* l
    for all ancestor a of l **do**
      with $\overline{C}_a$ *being the concept of the variable* a
      ▶ `Class:` C_l `SubClassOf:` $\overline{C}_a$
    end for
  end for

Algorithm 3 is based on Definitions 4 and 1.

The sequence s can be now classified by a DL reasoner according to the classes in O - or of any other ontology aligned to O. This process is demonstrated in Section 6 with a use case in the music domain.

The same approach can be used to convert the parse tree produced by algorithms such as Neural Network based Part of Speech tagging [2, 4] or Constituency Parsing [32, 40, 48].

5.3 Comparison with $\mathcal{SEQ}$

The work presented in [45] introduces $\mathcal{SEQ}$, an ontology pattern used to model sequence of elements using Description Logic and OWL. The method performs sequence classification by identifying sub-sequences through a subsumption relation: the sequence that is being classified subsumes a set of patterns (subsequences). Those patterns classify the sequence. The author shows how this method is only able to represent *safely-concatenable* CFG. A CFG is *safely-concatenable* if its productions are in the form $R \to t_1 \cdots t_n X$, with $X, R \in V$

and $t_1 \cdots t_n \in \Sigma$ [45]. V and Σ are defined as in Definition 1. Such restrictions prevent the representation of *self-embedding* grammars [45], which are grammars that contain productions of the type $R \to \alpha R \beta$, with $R \in V$ and $\alpha, \beta \in (V \cup \Sigma)$ [8]. Our proposal overcomes this limitation by directly reflecting the semantics of a production, as shown in Proof 4 of Theorem 1.

6 Experiments

In this section we apply our approach to the music domain [5] to perform the automatic analysis of harmonic progressions. Harmonic progressions are defined as sequences of chords, their analysis consists in assessing the underlying function of each chord [33]. Traditionally, it is performed by trained musicians since a deep knowledge and understanding of the music domain is required. The correctness depends on the taxonomy used and on the context in which the sequence is analysed (e.g. the genre).

In music theory, harmony is a well-researched area, and several taxonomies have been proposed to perform this task [36]. Most approaches classify each chord based on its tonal function, according to western musical theory, using CFG [11, 36] or Probabilistic CFG [17]. The implementation of these grammars can be problematic and relies on different techniques, such as Haskell datatypes in [11] or extensions to the definition of CFG in [17]. In [24] a CFG is used to detect sub-sequences, called *bricks*. *Bricks* are classes of chords sequences. Their combination defines new *bricks*. A similar approach is explored in [33], where the definition of *bricks* (called *idioms*) is performed through the use of a tree-like hierarchical ontology implemented using Object Oriented programming.

Our experiments are based on a subset of the rules implemented by [24], which we convert into an OWL ontology using Algorithm 1.

6.1 Grammar subset

The CFG defined in [24] can be formalized as $G_k = (V, \Sigma, R, S)$ where the set of variables V is the set of sub-sequences that will be extracted from a harmonic progression, Σ is the set of chords, R is the set of productions that maps each sequence to the corresponding set of chords. The starting S can be assigned to a special variable $V_s \in V$ such that $\forall\ t \in \Sigma\ \exists f \in R\ : s.t.\ f(V_s) = t$. To obtain a more tractable example, we extract a subset of the whole grammar G_k: we will only use the variables, terminals, and productions that are sufficient to analyze the tune *Blue Bossa* by *Dorham Kenny*. We then expand the grammar to include a few other productions that should not appear in the final analysis, to investigate how accurately the ontology reflects a grammar-based approach. A correspondence between the analysis of [24] and our results provides empirical evidences of the method correctness.

[5] The code of the experiments is available at https://github.com/n28div/CFGOwl under CC-BY License.

$$
\begin{aligned}
\text{OnOffMinorIV\_Cm} &\rightarrow \text{MinorOn\_Cm Off\_F} \\
\text{MinorOn\_Cm} &\rightarrow \text{C:min} \mid \text{C:minmaj7} \mid \text{C:min6} \mid \text{C:min7} \\
\text{Off\_F} &\rightarrow \text{F:7} \mid \text{F} \mid \text{F:maj} \mid \text{F:min} \mid \text{F:min7} \mid \text{F:minmaj7} \mid \text{F:dim7} \\
\text{SadCadence\_Cm} &\rightarrow \text{SadApproach\_Cm MinorOn\_Cm} \\
&\quad \mid \text{SadApproach\_Cm MinorOn\_Cm} \\
&\quad \mid \text{F:7(\#11) MinorPerfectCadence\_Cm} \\
\text{SadApproach\_Cm} &\rightarrow \text{D:hdim7 G:7} \\
\text{MinorPerfectCadence\_Cm} &\rightarrow \text{G:7 C:min7} \\
\text{StraightCadence\_Db} &\rightarrow \text{StraightApproach\_Db Db} \\
&\quad \mid \text{StraightApproach\_Db Db:maj7} \\
\text{StraightApproach\_Db} &\rightarrow \text{Eb:min7 StraightApproach\_C\_0} \\
&\quad \mid \text{Eb:min7 Ab:7} \\
\text{StraightApproach\_C\_0} &\rightarrow \text{Ab:7 C:7/Bb}
\end{aligned}
$$

Fig. 4: Productions of grammar $G_{k_1} \subseteq G_k$. Only productions are listed. The set of terminals and variables is the one used in the productions.

Figure 4 shows the formalization of the rules strictly needed to classify *Blue Bossa* as performed in [24]. Using algorithm 1 we convert the grammar in figure 4 into OWL. The resulting ontology contains 130 axioms. Using algorithm 2 we convert the chord annotations of *Blue Bossa*, taken from [24] into OWL. The resulting ontology contains a total of 29 axioms. By joining the two ontologies, we obtain a final ontology with a total of 159 axioms. The ontology correctly parses the sequence, as can be seen from table 1.

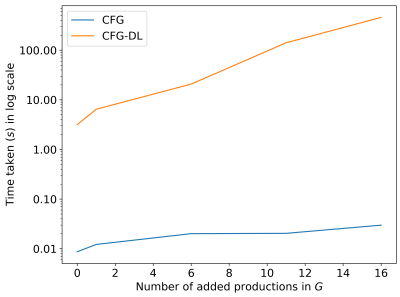
6.2 Reasoning complexity

As discussed in section 5.1, the computational complexity of parsing a sequence using a CFG-DL is exponential. On figure 5a we parse the song *Blue Bossa* using the grammar defined in Section 6.1. We then progressively add random productions to the grammar that do not affect the classification. At each iteration we add 5 new productions, which have a random number of right-hand sides sampled in the range $[1, 10]$. Each production is of type $R \rightarrow AB$ 80% of the time and $R \rightarrow t$ 20% of the time, to reflect the higher frequency of $R \rightarrow AB$ productions, especially when a CFG is expressed in CNF.

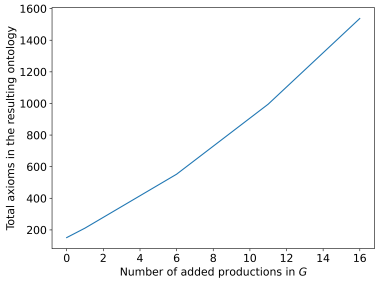
In Figure 5 empirical results from the described experiments are shown. Figure 5a shows how as productions are added to the grammar, the time complexity of CFG-DL increases exponentially. This is a consequence of the proportional increase of axioms as new productions are added (Figure 5b). When using the hybrid approach of Section 5.2 the computational complexity is much lower. In Figure 5a the time required to classify a sequence is significantly lower. All

Table 1: Parsing results for the harmonic progression of *Blue Bossa* using the grammar of figure 4. The class identified by [24] is represented in bold text

Chord	Inferred classes
C:min7	**OnOffMinorIV_Cm** VariableOne C:min7 MinorOn_Cm
F:min7	**OnOffMinorIV_Cm** VariableTwo F:min7 Off_F
D:hdim7	VariableOne SadApproach_Cm **SadCadence_Cm** D:hdim7
G:7	MinorPerfectCadence_Cm VariableTwo VariableOne G:7 SadApproach_Cm **SadCadence_Cm**
C:minmaj7	C:minmaj7 VariableTwo MinorOn_Cm **SadCadence_Cm**
Eb:min7	VariableOne StraightApproach_Db **StraightCadence_Db** Eb:min7
Ab:7	VariableTwo VariableOne StraightApproach_Db Ab:7 **StraightCadence_Db** StraightApproach_C_0
Db:maj7	VariableTwo Db:maj7 **StraightCadence_Db**
D:hdim7	VariableOne SadApproach_Cm **SadCadence_Cm** D:hdim7
G:7	MinorPerfectCadence_Cm VariableTwo VariableOne G:7 SadApproach_Cm **SadCadence_Cm**
C:minmaj7	C:minmaj7 VariableTwo MinorOn_Cm **SadCadence_Cm**



(a) Time taken (y axis, logarithmic) as random productions are added to the grammar (x axis). Parsing using DL is compared to the use of Earley parser on the corresponding CFG. The time taken by using the hybrid approach is 200% (3 orders of magnitude) than using DL parsing.



(b) Number of total axioms in the ontology (y axis) as random productions are added to the grammar (x axis). As more productions are added to the grammar, say N, roughly $10N$ axioms are inserted in the ontology. Since the computational complexity directly depends on the number of axioms in the ontology, the resulting CFG-DL is inefficient in real-world settings.

Fig. 5: Empirical results of the computational complexity when using a CFG-DL to parse the song *Blue Bossa*.

the experiments are executed using the Pellet reasoner [37] on a $2.4GHz$ *Intel i5-6300U* CPU and $8GB$ of RAM under regular computational load.

Even though the results of the two methods are indistinguishable, it is important to note that if the sequence is modified, Algorithm 3 need to be executed again, while a CFG-DL produced with Algorithm 1 would be able to classify the new element without any additional effort. We plan to address this aspect in future works, for instance by combining Algorithm 1 and Algorithm 3.

6.3 Subsequence classification

A complete understanding of the CFG is required to interpret Figure 4 and the results in Table 1. To obtain an higher interpretability, it is sufficient to expand the ontology produced by Algorithm 1 and increase the level of abstraction or by aligning other relevant ontologies. In the example of Figure 6, we can align the results with a domain-specific ontology, such as the Music Theory Ontology [34]. Differently from [33], the grammar and ontology definitions are decoupled in our approach.

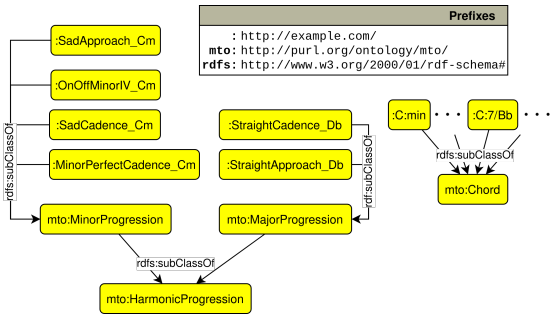


Fig. 6: Ontology imported by the ontology generated by algorithm 1.

The classification in Table 2 is obtained using the ontology of Figure 6. The results are arguably easier to interpret when compared to Table 1, without any update on the original CFG. By converting relevant grammars using Algorithm 1, a Knowledge Graph can be populated using the results of the parsing procedure. Additional classification can then be performed by aligning additional ontologies. For instance, to classify modal passages from a major to a minor progression (i.e. chord progression that transition from a major progression to a minor progression) as X it would be sufficient to define an axiom such as

▶((mto:MajorProgression and $\overline{V}_1$) or (mto:MinorProgression and $\overline{V}_2$)) SubClassOf: X

Table 2: Parsing results for the harmonic progression of *Blue Bossa* using the grammar of Figure 4 and importing Music Theory Ontology as shown in Figure 6.

Chord	Progression type
C:min7	Minor
F:min7	Minor
D:hdim7	Minor
G:7	Minor
C:minmaj7	Minor
Eb:min7	Major
Ab:7	Major
Db:maj	Major
D:hdim7	Minor
G:7	Minor
C:minmaj7	Minor

7 Conclusions

We present a novel approach to model a Context Free Grammar in Chomsky Normal Form using Description Logic. The computational complexity, as analysed in Sections 5.1 and 6.2, is too high to favour the usage of OWL for parsing sequences. However, as shown in Section 6.3, it enables the alignment of approaches based on Context Free Grammars with technologies typically used in the Semantic Web. Sequences can be represented and classified using OWL in an effective way by combining it with traditional parsing algorithms. This form of classification can be used for tasks such as the computation of similarity between two sequences. The inference of these similarities is of great use in the Music Information Retrieval field, where it is hard to define a similarity metric between two harmonic progression. The same approach, however, can be applied to other fields where sequences have been modeled using formal grammar, such as natural language processing [27], bio-informatics [9] and programming languages [10]. The hybrid approach using CFG and OWL ontologies allows a shift in the grammar modeling process: existing extensions, such as Combinatory Categorical Grammars (CCG) [38], have been proposed to transparently take into account the semantics of a sequence, along-side the syntactical aspects. It is possible to formalize a CCG in terms of DL, with a similar approach as the one presented in Section 4, and develop grammars whose semantic information is fueled by an expressive ontology.

Acknowledgements This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004746.

References

1. A. V. Aho and S. C. Johnson. LR parsing. *ACM Computing Surveys (CSUR)*, 6(2):99–124, 1974.
2. A. Akbik, D. Blythe, and R. Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th international conference on computational linguistics*, pages 1638–1649, 2018.
3. M. Baroni and C. Jacoboni. *Proposal for a grammar of melody : the Bach chorales*. Presses de l'Universite de Montreal Montreal, 1978.
4. B. Bohnet, R. T. McDonald, G. Simões, D. Andor, E. Pitler, and J. Maynez. Morphosyntactic Tagging with a Meta-BiLSTM Model over Context Sensitive Token Encodings. In I. Gurevych and Y. Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2642–2652. Association for Computational Linguistics, 2018.
5. R. Carrasco-Davis, G. Cabrera-Vives, F. Förster, P. A. Esté vez, P. Huijse, P. Protopapas, I. Reyes, J. Martínez-Palomera, and C. Donoso. Deep Learning for Image Sequence Classification of Astronomical Events. *Publications of the Astronomical Society of the Pacific*, 131(1004):108006, sep 2019.
6. T. Chen, R. Xu, Y. He, Y. Xia, and X. Wang. Learning User and Product Distributed Representations Using a Sequence Model for Sentiment Analysis. *IEEE Computational Intelligence Magazine*, 11(3):34–44, 2016.
7. N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956.
8. N. Chomsky. On certain formal properties of grammars. *Information and Control*, 2(2):137–167, 1959.
9. R. Damasevicius. Structural analysis of regulatory DNA sequences using grammar inference and support vector machine. *Neurocomputing*, 73(4-6):633–638, 2010.
10. C. Z. de Aguiar, R. de Almeida Falbo, and V. E. S. Souza. OOC-O: A reference ontology on object-oriented code. In A. H. F. Laender, B. Pernici, E. Lim, and J. P. M. de Oliveira, editors, *Conceptual Modeling - 38th International Conference, ER 2019, Salvador, Brazil, November 4-7, 2019, Proceedings*, volume 11788 of *Lecture Notes in Computer Science*, pages 13–27. Springer, 2019.
11. W. B. De Haas, J. P. Magalhães, F. Wiering, and R. C. Veltkamp. Automatic functional harmonic analysis. *Computer Music Journal*, 37(4):37–53, 2013.
12. N. Drummond, A. L. Rector, R. Stevens, G. Moulton, M. Horridge, H. Wang, and J. Seidenberg. Putting OWL in order: Patterns for sequences in OWL. In B. C. Grau, P. Hitzler, C. Shankey, and E. Wallace, editors, *Proceedings of the OWLED*06 Workshop on OWL: Experiences and Directions, Athens, Georgia, USA, November 10-11, 2006*, volume 216 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.
13. J. Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970.
14. R. C. B. G. Edward Barton and E. S. Ristad. Computational complexity and natural language. *Journal of Linguistics*, 24(2):573–575, 1987.
15. R. Giegerich. Introduction to stochastic context free grammars. *RNA Sequence, Structure, and Function: Computational and Bioinformatic Methods*, pages 85–106, 2014.
16. B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang. HermiT: an OWL 2 reasoner. *Journal of Automated Reasoning*, 53(3):245–269, 2014.

17. D. Harasim, M. Rohrmeier, and T. J. O'Donnell. A Generalized Parsing Framework for Generative Models of Harmonic Syntax. In *ISMIR*, pages 152–159, 2018.
18. C. Harte, M. B. Sandler, S. A. Abdallah, and E. Gómez. Symbolic Representation of Musical Chords: A Proposed Syntax for Text Annotations. In *ISMIR*, volume 5, pages 66–71, 2005.
19. P. Hitzler, A. Gangemi, K. Janowicz, A. Krisnadhi, and V. Presutti, editors. *Ontology Engineering with Ontology Design Patterns - Foundations and Applications*, volume 25 of *Studies on the Semantic Web*. IOS Press, 2016.
20. J. E. Hopcroft, R. Motwani, and J. D. Ullman. Introduction to automata theory, languages, and computation. *Acm Sigact News*, 32(1):60–65, 2001.
21. M. Horridge, N. Drummond, J. Goodwin, A. L. Rector, R. Stevens, and H. Wang. The Manchester OWL syntax. In *OWLed*, volume 216, 2006.
22. I. Horrocks, B. Parsia, and U. Sattler. OWL 2 web ontology language direct semantics. *World Wide Web Consortium*, pages 42–65, 2012.
23. I. Horrocks and P. F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In *International semantic web conference*, pages 17–29. Springer, 2003.
24. R. Keller, A. Schofield, A. Toman-Yih, Z. Merritt, and J. Elliott. Automating the explanation of jazz chord progressions using idiomatic analysis. *Computer Music Journal*, 37(4):54–69, 2013.
25. A. Krisnadhi, F. Maier, and P. Hitzler. OWL and rules. In *Reasoning Web International Summer School*, pages 382–415. Springer, 2011.
26. S. Kumar. Gene Sequence Classification Using K-mer Decomposition and Soft-Computing-Based Approach. In T. K. Sharma, C. W. Ahn, O. P. Verma, and B. K. Panigrahi, editors, *Soft Computing: Theories and Applications*, pages 181–186, Singapore, 2021. Springer Singapore.
27. S. Lawrence, C. Giles, and S. Fong. Natural language grammatical inference with recurrent neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 12(1):126–140, 2000.
28. J. Lei, Q. Zhang, J. Wang, and H. Luo. BERT Based Hierarchical Sequence Classification for Context-Aware Microblog Sentiment Analysis. In T. Gedeon, K. W. Wong, and M. Lee, editors, *Neural Information Processing*, pages 376–386, Cham, 2019. Springer International Publishing.
29. F. Lerdahl and R. Jackendoff. *A generative theory of tonal music*. The MIT Press, Cambridge. MA, 1983.
30. G. Lo Bosco and M. A. Di Gangi. Deep Learning Architectures for DNA Sequence Classification. In A. Petrosino, V. Loia, and W. Pedrycz, editors, *Fuzzy Logic and Soft Computing Applications*, pages 162–171, Cham, 2017. Springer International Publishing.
31. S. Madhusudhan and G. Chowdhary. Deepsrgm - Sequence classification and ranking in Indian classical music with deep learning. In A. Flexer, G. Peeters, J. Urbano, and A. Volk, editors, *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, pages 533–540. International Society for Music Information Retrieval, 2019.
32. K. Mrini, F. Dernoncourt, Q. H. Tran, T. Bui, W. Chang, and N. Nakashole. Rethinking self-attention: Towards interpretability in neural parsing. In T. Cohn, Y. He, and Y. Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 731–742. Association for Computational Linguistics, 2020.

33. F. Pachet. Computer analysis of jazz chord sequence: is solar a blues?, 2000.
34. S. M. Rashid, D. De Roure, and D. L. McGuinness. A music theory ontology. In *Proceedings of the 1st International Workshop on Semantic Applications for Audio and Music*, pages 6–14, 2018.
35. E. Rivas and S. R. Eddy. The language of RNA: a formal grammar that includes pseudoknots. *Bioinform.*, 16(4):334–340, 2000.
36. M. Rohrmeier. Towards a generative syntax of tonal harmony. *Journal of Mathematics and Music*, 5(1):35–53, 2011.
37. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.
38. M. Steedman and J. Baldridge. Combinatory categorial grammar. *Non-Transformational Syntax: Formal and Explicit Models of Grammar. Wiley-Blackwell*, pages 181–224, 2011.
39. M. J. Steedman. A Generative Grammar for Jazz Chord Sequences. *Music Perception: An Interdisciplinary Journal*, 2(1):52–77, 1984.
40. Y. Tian, Y. Song, F. Xia, and T. Zhang. Improving Constituency Parsing with Span Attention. In T. Cohn, Y. He, and Y. Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 1691–1703. Association for Computational Linguistics, 2020.
41. D. Tidhar. *A hierarchical and deterministic approach to music grammars and its application to unmeasured preludes*. PhD thesis, Berlin Institute of Technology, 2005.
42. S. Tojo, Y. Oka, and M. Nishida. Analysis of chord progression by HPSG. In *Proceedings of the 24th IASTED International Conference on Artificial Intelligence and Applications*, AIA'06, page 305–310, USA, 2006. ACTA Press.
43. L. G. Valiant. General context-free recognition in less than cubic time. *Journal of computer and system sciences*, 10(2):308–315, 1975.
44. D. Weininger. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.
45. J. Wissmann. *Chord Sequence patterns in OWL*. PhD thesis, City University London, 2012.
46. K. Wolstencroft, R. Stevens, and V. Haarslev. Applying OWL Reasoning to Genomic Data. In C. J. O. Baker and K.-H. Cheung, editors, *Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences*, pages 225–248, Boston, MA, 2007. Springer US.
47. Z. Xing, J. Pei, and E. Keogh. A brief survey on sequence classification. *SIGKDD Explor. Newsl.*, 12(1):40–48, Nov. 2010.
48. K. Yang and J. Deng. Strongly incremental constituency parsing with graph neural networks. *Advances in Neural Information Processing Systems*, 33:21687–21698, 2020.
49. M. Young-Lai. Grammar inference. In L. LIU and M. T. ÖZSU, editors, *Encyclopedia of Database Systems*, pages 1256–1260, Boston, MA, 2009. Springer US.