

# Logics for complexity classes

Vladimir Naidenko\*  
 Institute of Mathematics  
 National Academy of Sciences of Belarus  
 ul. Surganova 11, Minsk 220012, Belarus

September 8, 2018

## Abstract

A new syntactic characterization of problems complete via Turing reductions is presented. General canonical forms are developed in order to define such problems. One of these forms allows us to define complete problems on ordered structures, and another form to define them on unordered non-Aristotelian structures. Using the canonical forms, logics are developed for complete problems in various complexity classes. Evidence is shown that there cannot be any complete problem on Aristotelian structures for several complexity classes. Our approach is extended beyond complete problems. Using a similar form, a logic is developed to capture the complexity class  $NP \cap coNP$  which very likely contains no complete problem.

*Keywords:* theory of computation, computational complexity, Turing reduction, completeness, descriptive complexity

## 1 Introduction

Since 1974, descriptive complexity characterizes computational complexity in terms of logical languages. Fagin [6] first shown that the complexity class NP coincides with the set of problems expressible in second order existential ( $SO\exists$ ) logic. Stockmeyer [19] extended Fagin's result to the polynomial-time hierarchy (PH) characterized by second order logic. Further researches revealed logical characterizations for various complexity classes [11, 17]. Immerman presented a lot of results on descriptive complexity in his diagram [11] whose fragment is shown in Figure 1.

Medina and Immerman [14] addressed the following question. What is it about a  $SO\exists$ -sentence that makes the expressed property NP-complete? They answered this question in part. Namely, a necessary and sufficient syntactic

---

\*E-mail: [naidenko@open.by](mailto:naidenko@open.by)

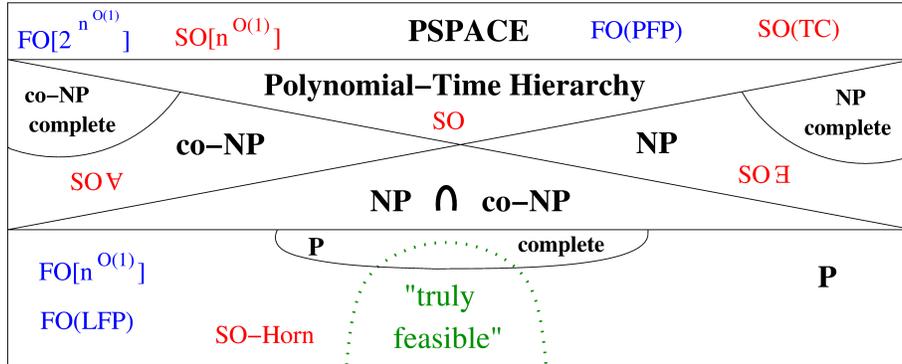


Figure 1: The World of Descriptive and Computational Complexity from P to PSPACE.

condition is provided for a  $SO\exists$ -sentence to represent a NP-complete problem on ordered structures. In addition, the problem is NP-complete via first-order projections (fops). The condition is expressed in some canonical form based on the CLIQUE problem. This canonical form provides a syntactic tool for showing NP-completeness: if a  $SO\exists$ -sentence is of the form, then the problem defined by this  $SO\exists$ -sentence proves to be NP-complete via fops.

Bonet and Borges [3] generalized Medina-Immerman's canonical form in two directions. First, Bonet-Borges's canonical form works for various complexity classes on ordered structures, and not just NP. Second, the form can be given in terms of any complete problem in the class, and not just CLIQUE in case of NP.

However, all the above mentioned forms have a restricted application. No fops are known for the vast majority of complete problems. For example, in the complexity class NP, only up to fifty NP-complete problems are known to be complete via fops [3, 14]. Moreover, only five numerical NP-complete problems among thousands of such ones are known to be complete via fops [3]. Besides, these forms define complete problems merely on ordered structures, and no canonical form was known till now to define complete problems on unordered structures.

As shown in Figure 1, there are complexity classes such as NP-complete problems, coNP-complete problems, P-complete problems, and  $NP \cap coNP$  for which no logics were known till now. In addition, for the complexity classes of PSPACE-complete problems and NL-complete problems which are not depicted in Immerman's diagram, no logics were known as well. The purpose of our research is to develop logics for these classes.

We will consider problems complete via Turing reductions, and not just fops. A new canonical form will be presented to easily define any complete problem on ordered structures in the following complexity classes: NL, P, coNP, NP, and PSPACE (no matter whether it is complete via fops or not). In that way, we

will completely answer the above mentioned question of Medina and Immerman. Moreover, using the form, logics will be developed for complete problems on ordered structures in these complexity classes.

We will show an evidence that there cannot be any complete problem on Aristotelian structures in P, coNP, NP, and PSPACE.

A new canonical form will be presented to define any complete problem on unordered non-Aristotelian structures in the following complexity classes: coNP, NP, and PSPACE. Using this form, logics will be developed for complete problems on unordered non-Aristotelian structures in these complexity classes.

Furthermore, we will extend our approach beyond complete problems. By means of a very similar canonic form for defining problems in the complexity class  $\text{NP} \cap \text{coNP}$ , a logic will be developed to capture  $\text{NP} \cap \text{coNP}$  which very likely contains no complete problem.

## 2 Preliminaries

We specify some of the notations and conventions used throughout this paper. We denote by  $\log x$  the logarithm of  $x$  to the base 2. For a real number  $x$ , we denote by  $\lfloor x \rfloor$  the greatest  $n \in \mathbb{Z}$  subject to  $n \leq x$ . We denote by  $\tilde{x}$  the prefix-free encoding [9] of a nonnegative integer  $x$ . The length of a string  $w$  is denoted by  $|w|$ . For strings  $w_1$  and  $w_2$ , we denote by  $w_1w_2$  their concatenation. For a string  $w$  and a nonnegative integer  $x$ , we denote by  $w^x$  the concatenation  $\underbrace{w \dots w}_{x \text{ times}}$  (if  $x = 0$ , then this concatenation is interpreted as the empty string). By  $w[i]$  we denote the  $i$ -th bit of a nonempty binary string  $w$  (for definiteness, we will assume that numbering the bits of any nonempty binary string starts with 1 from left to right). For a natural number  $x$ , we denote by  $\ell(x)$  the length of  $x$  represented in binary, i.e.  $\ell(x) = \lfloor \log x \rfloor + 1$ . By  $\ell^{(k)}$  we mean the function composition  $\underbrace{\ell \circ \dots \circ \ell}_{k \text{ times}}$ .

We use notations and definitions of finite model theory as stated in [8, 11]. For convenience and without loss of generality, we will consider vocabularies without constant symbols and without function symbols. So, a vocabulary is a finite set  $\tau = \{R_1^{a_1}, \dots, R_m^{a_m}\}$  of relation symbols of specified arities. A  $\tau$ -structure is a tuple  $A = (|A|, R_1^A, \dots, R_m^A)$ , where  $|A|$  is a nonempty set, called the universe of  $A$ , and each  $R_i^A$  is a relation on  $A$  such that  $\text{arity}(R_i^A) = a_i$ ,  $1 \leq i \leq m$ . A  $\tau$ -structure is ordered if one of the symbols of its vocabulary  $\tau$  is  $<$ , interpreted as a linear order on the universe.

A finite  $\tau$ -structure is a  $\tau$ -structure  $A$  whose universe  $|A|$  is a finite set. We will assume that the universe of every finite  $\tau$ -structure is an initial segment  $\{0, \dots, n-1\}$  of nonnegative integers, where  $n > 1$ . The notation  $\|A\|$  denotes the cardinality of the universe of  $A$ . Only finite structures will be considered throughout this paper. Therefore, we will omit the word ‘finite’, when referring to the structures in what follows.

Let us suppose that  $A = (|A|, R_1^A, \dots, R_m^A)$  and  $B = (|B|, R_1^B, \dots, R_m^B)$  are

two  $\tau$ -structures. An isomorphism between  $A$  and  $B$  is a mapping  $h : |A| \rightarrow |B|$  that satisfies the following two conditions:

- 1)  $h$  is a bijection.
- 2) For every relation symbol  $R_i$ ,  $1 \leq i \leq m$ , of arity  $t$  and for every  $t$ -tuple  $(a_1, \dots, a_t)$  of elements in  $|A|$ , we have  $R_i^A(a_1, \dots, a_t)$  if and only if  $R_i^B(h(a_1), \dots, h(a_t))$ .

By a model class we mean a set  $K$  of  $\tau$ -structures of a fixed vocabulary  $\tau$  that is closed under isomorphism, i.e. if  $A \in K$  and  $A$  is isomorphic to  $B$ , then  $B \in K$  as well. For a vocabulary  $\tau$ , we by  $\text{STRUC}[\tau]$  denote the model class of all  $\tau$ -structures.

Let  $\mathcal{L}$  denote a logic. For every vocabulary  $\tau$ , the language  $\mathcal{L}(\tau)$  is the recursive set of all well-formed sentences (whose elements are called  $\mathcal{L}(\tau)$ -sentences) with the symbols of  $\tau$  and with the symbols predefined for the logic  $\mathcal{L}$ . For example, for first order (FO) logic,  $\text{FO}(\tau)$  is the set of all first order sentences with the symbols of  $\tau$ , numerical relational symbols:  $=, \neq$ , logical connectives:  $\wedge, \vee, \neg$ , variables:  $x, y, z, \dots$  with or without subscripts, quantifiers:  $\exists, \forall$ , and brackets:  $(, ), [, ]$ . All the predefined symbols have the standard interpretations. Also, for  $\text{SO}\exists$  logic,  $\text{SO}\exists(\tau)$  denotes the set of all second order sentences of the form  $\exists Q_1 \dots \exists Q_i \varphi$ , where  $\varphi$  is a  $\text{FO}(\tau \cup \{Q_1^{a_1}, \dots, Q_i^{a_i}\})$ -sentence.

In addition,  $\models$  is a binary relation between  $\mathcal{L}(\tau)$ -sentences and  $\tau$ -structures, so that for each  $\mathcal{L}(\tau)$ -sentence  $\Gamma$ , the set  $\{A \in \text{STRUC}[\tau] \mid A \models \Gamma\}$  denoted by  $\text{MOD}[\Gamma]$  is closed under isomorphism. Also, we say that a  $\mathcal{L}(\tau)$ -sentence  $\Gamma$  defines a model class  $K$  if  $K = \text{MOD}[\Gamma]$ . We call a  $\mathcal{L}(\tau)$ -sentence  $\Gamma$  logically valid if  $\text{STRUC}[\tau] = \text{MOD}[\Gamma]$ . We say that two  $\mathcal{L}(\tau)$ -sentences  $\Lambda$  and  $\Gamma$  are logically equivalent if  $\text{MOD}[\Lambda] = \text{MOD}[\Gamma]$ .

By  $\Gamma(\psi)$  we denote a sentence  $\Gamma$  such that  $\psi$  occurs in  $\Gamma$  as a subformula. Then,  $\Gamma[\psi/\xi]$  denotes the sentence obtained by replacing each occurrence of  $\psi$  by  $\xi$  in  $\Gamma$ .

In order to measure the computational complexity of problems on structures, we need to represent structures by binary strings to be used as inputs for Turing machines. Moreover, simple encodings can be used to represent any arbitrary objects (integers, sentences, Turing machines, etc.) as binary strings.

For an object  $Z$ , we will use  $\langle Z \rangle$  to denote some binary representation of  $Z$ . For example, if  $Z$  is a natural number, then  $\langle Z \rangle$  denotes its usual binary representation. If  $Z$  is a sentence, then  $\langle Z \rangle$  denotes the binary representation of Gödel number of  $Z$ . If  $Z$  is a Turing machine, then  $\langle Z \rangle$  also denotes the binary representation of Gödel number of  $Z$ . If  $Z$  is a  $\tau$ -structure, then  $\langle Z \rangle$  denotes the binary encoding  $\text{bin}(Z)$  as stated in [11].

We will characterize a model class of  $\text{STRUC}[\tau]$  for some fixed vocabulary  $\tau$  as a complexity theoretic problem. Let  $\mathcal{L}$  be a logic,  $\mathcal{N}$  a complexity class, and  $\tau$  a vocabulary. We say that  $\mathcal{L}$  captures  $\mathcal{N}$  on  $\text{STRUC}[\tau]$  if the following two conditions are satisfied:

- 1) For every  $\mathcal{L}(\tau)$ -sentence  $\Gamma$ , the model class  $\text{MOD}[\Gamma]$  belongs to  $\mathcal{N}$ .

- 2) For every model class  $K \subseteq \text{STRUC}[\tau]$  in  $\mathcal{N}$ , there exists a  $\mathcal{L}(\tau)$ -sentence  $\Gamma$  that defines  $K$ .

Also, we say that  $\mathcal{L}$  captures  $\mathcal{N}$  on all (ordered) structures if  $\mathcal{L}$  captures  $\mathcal{N}$  on  $\text{STRUC}[\tau]$ , for every vocabulary  $\tau$  (containing  $<$ ).

We need the following definition of oracle Turing machines adapted for structures [11, 12]. An oracle Turing machine is a Turing machine with a two-way read only input tape, a two-way read-write storage tape, and a one-way write only oracle tape. Oracle Turing machines have special states: *ACC*, *QUE*, *YES*, and *NO*. The state *ACC* is the accepting state. That is, a  $\tau$ -structure  $A$  is accepted by a Turing machine  $T$  if and only if  $T$  on the input  $\langle A \rangle$  halts in state *ACC*. The state *QUE* is the query state. In each state except *QUE* the machine may write a symbol onto the oracle tape. In state *QUE* the machine goes into state *YES* if the string written on the oracle tape is a member of the oracle set, otherwise it enters state *NO*. In moving from state *QUE* to *YES* or *NO* no other action is taken except to erase the oracle tape. By  $T^B$  we denote the oracle Turing machine  $T$  equipped with an oracle for the set  $B$  of some  $\tau$ -structures. For example, the string  $\langle A \rangle$  encoding a  $\tau$ -structure  $A$  is written on the oracle tape when  $T^B$  enters the query state *QUE*. At the next step,  $T^B$  goes into either *YES* if  $A \in B$ , or *NO* otherwise; the oracle tape is (magically) erased.

We also need the following definitions of Turing machines with bounded resources.

A Turing machine  $T$  is polynomial time clocked if the code of  $T$  contains a natural number  $c$  such that  $(n + 2)^c$  is an upper bound for the running time of  $T$  on inputs of length  $n$ .

A Turing machine  $T$  is logarithmic space (logspace) bounded if the code of  $T$  contains a natural number  $c$  such that  $c \log(n + 2)$  is an upper bound for the memory space of  $T$ 's storage tape on inputs of length  $n$ .

Let us by  $\mathcal{N}$  mean some complexity class. For short, we will call an oracle Turing machine  $\mathcal{N}$ -specific if it is of the same type as the machines that usually realize Turing reductions of problems in  $\mathcal{N}$ . So, in case of  $\mathcal{N} \in \{\text{coNP}, \text{NP}, \text{PSPACE}\}$ , by  $\mathcal{N}$ -specific Turing machines we mean polynomial time clocked oracle Turing machines. In case of  $\mathcal{N} \in \{\text{NL}, \text{P}\}$ , by  $\mathcal{N}$ -specific Turing machines we mean logspace bounded oracle Turing machines.

Let us recall the following definitions of Turing reducibility and completeness. Given a complexity class  $\mathcal{N}$ , for two sets  $A$  and  $B$  in  $\mathcal{N}$ , we say that  $A$  is Turing reducible to  $B$  if there exists a  $\mathcal{N}$ -specific Turing machine  $T^B$  to recognize the set  $A$ . A set  $A \in \mathcal{N}$  is called  $\mathcal{N}$ -complete (via Turing reductions) if  $X$  is Turing reducible to  $A$  for all  $X \in \mathcal{N}$ .

### 3 Aristotelian structures and completeness

In this section, let  $\mathcal{N}$  denote one of the following complexity classes:  $\text{P}$ ,  $\text{coNP}$ ,  $\text{NP}$ , or  $\text{PSPACE}$ . We will present a vocabulary  $\tau$  such that the ex-

istence of a  $\mathcal{N}$ -complete problem defined on the  $\tau$ -structures is unlikely. Let  $\tau = \{R_1^1, \dots, R_m^1\}$  be a fixed vocabulary that contains only the symbols of monadic predicates. We call such a vocabulary Aristotelian [13]. Let  $A = (|A|, R_1^A, \dots, R_m^A)$  be an arbitrary  $\tau$ -structure. We also call such a  $\tau$ -structure Aristotelian.

Let us consider in detail the binary encoding  $\langle A \rangle$  of an Aristotelian  $\tau$ -structure  $A$ . This encoding  $\langle A \rangle$  represents the concatenation of some binary strings:  $w_1^A \dots w_m^A$ , where  $n = \|A\|$ ;  $w_j^A \in \{0, 1\}^n$ ,  $1 \leq j \leq m$ . Each string  $w_j^A$ ,  $1 \leq j \leq m$ , is defined as follows: either  $w_j^A[i+1] = 1$  if  $R_j^A(i)$ , or  $w_j^A[i+1] = 0$  otherwise, for every  $0 \leq i \leq n-1$ .

Let us show that any Aristotelian  $\tau$ -structure  $A$  can be encoded in a more condensed form as compared with the encoding  $\langle A \rangle$ . We write in lexicographic order the sequence  $v_1, \dots, v_{2^m}$  of all binary strings in  $\{0, 1\}^m$ . Let us introduce the following characteristic function  $\chi^A : \{0, \dots, n-1\} \times \{1, \dots, 2^m\} \rightarrow \{0, 1\}$  for the  $\tau$ -structure  $A$ :

$$\chi^A(i, j) \triangleq \begin{cases} 1 & \text{if } v_j = w_1^A[i+1]w_2^A[i+1] \dots w_m^A[i+1]; \\ 0 & \text{otherwise.} \end{cases}$$

Then, we define a mapping  $benc : \text{STRUC}[\tau] \rightarrow \{0, 1\}^*$  as follows:

$$benc(A) \triangleq 1v_1\tilde{n}_1v_2\tilde{n}_2 \dots v_{2^m}\tilde{n}_{2^m}$$

where each  $n_j$ ,  $1 \leq j \leq 2^m$ , is equal to  $\sum_{i=0}^{n-1} \chi^A(i, j)$ .

Let us estimate the length of  $benc(A)$ . Note that  $|benc(A)| \leq 1 + 2^m(m + \log n + 2 \log \log n + 7)$ , where  $n = \|A\|$ . Therefore, the length of  $benc(A)$  grows logarithmically in  $n$  since  $m$  is a constant, while the length of  $\langle A \rangle$  grows polynomially in  $n$ . That is,  $benc(A)$  encodes  $A$  in a more condensed form as compared with  $\langle A \rangle$ .

Note that any Aristotelian  $\tau$ -structure  $A$  can be also encoded as a unary string. We denote by  $uenc(A)$  the unary string  $1^{benc(A)}$ , where  $benc(A)$  is interpreted here as a number (represented in binary). Then,  $uenc(A)$  can be just considered as the unary encoding of  $A$ . Since  $|uenc(A)| \leq 2^{|benc(A)|} - 1$ , we have  $|uenc(A)| < 2^{(m+7)2^m+1} \cdot (n \log^2 n)^{2^m}$ , where  $n = \|A\|$ . That is, the length of  $uenc(A)$  grows polynomially in  $n$  since  $m$  is a constant.

We can unambiguously recover  $benc(A)$  from  $uenc(A)$  as  $benc(A) = \langle |uenc(A)| \rangle$  since  $benc(A)$  always starts with 1. However, we cannot unambiguously recover  $A$  from  $benc(A)$  or  $uenc(A)$ . Nevertheless, the function  $uenc$  defines the isomorphism property between Aristotelian  $\tau$ -structures, which is shown by the following lemma.

**Lemma 1** *Any two Aristotelian  $\tau$ -structures  $A$  and  $B$  are isomorphic if and only if  $uenc(A) = uenc(B)$ .*

PROOF. Let  $h : |A| \rightarrow |B|$  be an isomorphism between the  $\tau$ -structures  $A$  and  $B$ . Note that the statement  $\chi^A(a, j) = \chi^B(h(a), j)$  holds for any  $a \in |A|$

and  $1 \leq j \leq 2^m$ . It follows that  $benc(A) = benc(B)$  and, therefore,  $uenc(A) = uenc(B)$ .

Suppose that  $uenc(A) = uenc(B)$ . Then, we have  $benc(A) = benc(B)$ . We will show that the equality  $benc(A) = benc(B)$  implies an isomorphism between  $A$  and  $B$ . We denote by  $\prec_A$  (resp.  $\prec_B$ ) the following linear order on  $|A|$  (resp.  $|B|$ ). For any  $x_1, x_2 \in |A|$  (resp.  $y_1, y_2 \in |B|$ ), we say that  $x_1 \prec_A x_2$  (resp.  $y_1 \prec_B y_2$ ) if there exist natural numbers  $j_1$  and  $j_2$  such that one of the two conditions is satisfied:

- 1)  $1 \leq j_1 < j_2 \leq 2^m$  and  $\chi^A(x_1, j_1) = \chi^A(x_2, j_2) = 1$  (resp.  $\chi^B(y_1, j_1) = \chi^B(y_2, j_2) = 1$ ).
- 2)  $1 \leq j_1 = j_2 \leq 2^m$ ,  $\chi^A(x_1, j_1) = \chi^A(x_2, j_2) = 1$  (resp.  $\chi^B(y_1, j_1) = \chi^B(y_2, j_2) = 1$ ), and  $x_1 < x_2$  (resp.  $y_1 < y_2$ ).

Let us write in orders  $\prec_A$  and  $\prec_B$  the sequences  $a_1 \prec_A a_2 \prec_A \dots \prec_A a_n$  and  $b_1 \prec_B b_2 \prec_B \dots \prec_B b_n$  of all elements in  $|A|$  and  $|B|$  respectively, where  $n = \|A\|$  (note that  $\|A\| = \|B\|$  since  $benc(A) = benc(B)$ ). Using these sequences, we define a function  $g : |A| \rightarrow |B|$  as follows. For any  $a \in |A|$ , we have  $g(a) \triangleq b_j$ , where  $j$  such that  $a_j = a$ .

Note that the equality  $\chi^A(a, j) = \chi^B(g(a), j)$  holds for any  $a \in |A|$  and  $1 \leq j \leq 2^m$ . In other words, for every  $a \in |A|$ , we have  $R_i^A(a)$  if and only if  $R_i^B(g(a))$ ,  $1 \leq i \leq m$ . That is, this bijection  $g$  represents an isomorphism between  $A$  and  $B$ . This concludes the proof of the lemma.  $\square$

Then, the following theorem holds.

**Theorem 2** *Let  $\tau = \{R_1^1, \dots, R_m^1\}$  be a fixed Aristotelian vocabulary. If there exists a  $\mathcal{N}$ -complete model class of  $\tau$ -structures, then there exists a unary  $\mathcal{N}$ -complete set.*

**PROOF.** Let  $L$  be some  $\mathcal{N}$ -complete model class of Aristotelian  $\tau$ -structures. Let us consider the set  $M = \{u \in 1^* \mid (\exists A \in \text{STRUC}[\tau])[u = uenc(A) \wedge A \in L]\}$ .

First, we will reduce  $L$  to  $M$  via a  $\mathcal{N}$ -specific Turing machine  $T^M$ . Given a  $\tau$ -structure  $A$ , this machine  $T^M$  transforms the input  $\langle A \rangle$  into the binary string  $benc(A)$  on the storage tape. Then,  $T^M$  rewrites  $benc(A)$  in the unary encoding  $uenc(A)$  onto the oracle tape and enters state *QUE*. Next, if  $T^M$  goes into state *YES* (i.e.  $uenc(A) \in M$ ), then  $T^M$  immediately moves to state *ACC* (i.e.  $T^M$  accepts  $A$ ). Otherwise,  $T^M$  rejects  $A$ . Note that this reduction requires at most logarithmic space since  $|benc(A)|$  grows logarithmically in  $\|A\|$ . Therefore,  $T^M$  is indeed  $\mathcal{N}$ -specific.

Second, we will show that  $M$  belongs to the complexity class  $\mathcal{N}$ . Let an arbitrary string  $u \in 1^*$  be given. Let us decide whether  $u \in M$  or not. We transform  $u$  into the binary string  $w = \langle |u| \rangle$ . Then, we verify in polynomial time whether  $w$  is of the form  $1v_1\tilde{n}_1v_2\tilde{n}_2\dots v_{2^m}\tilde{n}_{2^m}$  or not, where  $\sum_{l=1}^{2^m} n_l > 1$ . If this is not the case, then  $u \notin M$ . Otherwise, we construct a  $\tau$ -structure  $A = (|A|, R_1^A, \dots, R_m^A)$  in the binary encoding  $\langle A \rangle$  by means of the following algorithm:

- 1) Set  $i := 1, j := 1, k := 1, n := \sum_{l=1}^{2^m} n_l$ .
- 2) Set the binary strings:  $w_1^A := 0^n, \dots, w_m^A := 0^n$ .
- 3) If  $j > 2^m$ , then stop (the  $\tau$ -structure  $A$  is formed in the binary encoding  $\langle A \rangle = w_1^A \dots w_m^A$ ). Otherwise, go to step 4.
- 4) If  $n_j = 0$ , then set  $j := j + 1$  and go to step 3. Otherwise, go to step 5.
- 5) For every  $1 \leq r \leq m$ , set  $w_r^A[i] := 1$  if  $v_j[r] = 1$ .
- 6) Set  $i := i + 1$  and  $k := k + 1$ . If  $k > n_j$ , then set  $k := 1, j := j + 1$  and go to step 3. Otherwise, go to step 5.

Since the set  $L$  is closed under isomorphism, the string  $u$  belongs to  $M$  if and only if this  $\tau$ -structure  $A$  is in  $L$ . It is obvious that constructing the  $\tau$ -structure  $A$  from the input  $u$  takes polynomial time. Hence,  $M$  belongs to  $\mathcal{N}$ . Thus, the set  $M$  is  $\mathcal{N}$ -complete. This concludes the proof of the theorem.  $\square$

By Theorem 2, it is unlikely that there is a  $\mathcal{N}$ -complete model class of Aristotelian structures. For example, the existence of a NP-complete unary language implies  $P = NP$  (see [2]). Also, the existence of a P-complete sparse language under many-one logspace reduction implies  $L = P$  (see [4]). Therefore, we will consider  $\mathcal{N}$ -complete problems on non-Aristotelian structures in what follows.

## 4 Canonical forms for complete problems

### 4.1 Canonical form for complete problems on ordered structures

We will introduce a canonical form for  $\mathcal{N}$ -complete problems on ordered structures. We denote by  $\sigma_{<}$  the vocabulary  $\{R^1, <\}$ . We assume that first order logic is extended with the numerical predicate BIT which is defined in [11]. Let  $\mathcal{L}_{\mathcal{N}}$  be a logic capturing  $\mathcal{N}$  on  $\text{STRUC}[\sigma_{<}]$ , and  $\Upsilon(R(x))$  a distinguished  $\mathcal{L}(\sigma_{<})$ -sentence defining some  $\mathcal{N}$ -complete model class. In this section, we by  $\mathcal{N}$  mean one of the following complexity classes: NL, P, coNP, NP, or PSPACE. For definiteness, we assume that  $\mathcal{L}_{\text{NL}}, \mathcal{L}_{\text{P}}, \mathcal{L}_{\text{coNP}}, \mathcal{L}_{\text{NP}}$ , and  $\mathcal{L}_{\text{PSPACE}}$  stand for the following logics: FO(TC), FO(LFP), SO $\forall$ , SO $\exists$ , and SO(PFP) respectively.

We need to be able to construct sentences defining  $\mathcal{N}$ -complete problems for various vocabularies containing  $<$ , using an operator over  $\Upsilon(R(x))$ . Let  $\tau_{<} = \{R_1^{a_1}, \dots, R_m^{a_m}, <\}$  be a fixed arbitrary vocabulary containing  $<$ . We define the operator  $T_{\tau_{<}}$  for mapping  $\Upsilon(R(x))$  to a  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentence as follows. If  $a_1 = 1$ , then  $T_{\tau_{<}}(\Upsilon(R(x))) \triangleq \Upsilon[R(x)/R_1(x)]$ . If  $a_1 > 1$ , then  $T_{\tau_{<}}(\Upsilon(R(x))) \triangleq \Upsilon[R(x)/\underbrace{\exists y R_1(y, \dots, y, x)}_{a_1 - 1 \text{ times}}]$ , where  $y$  is a new variable (we assume that  $y$  does not occur anywhere in  $\Upsilon(R(x))$ ). For short, let  $\Upsilon_{\tau_{<}}$  denote  $T_{\tau_{<}}(\Upsilon(R(x)))$ .

Let us show that the  $\mathcal{L}(\tau_{<})$ -sentence  $\Upsilon_{\tau_{<}}$  defines a  $\mathcal{N}$ -complete problem. We will reduce  $\text{MOD}[\Upsilon(R(x))]$  to  $\text{MOD}[\Upsilon_{\tau_{<}}]$ . For this purpose, we introduce a  $\mathcal{N}$ -specific Turing machine  $T^{\text{MOD}[\Upsilon_{\tau_{<}}]}$  that recognizes the model class  $\text{MOD}[\Upsilon(R(x))]$  as follows.

Let the binary string  $\langle A \rangle$  encoding some  $\sigma_{<}$ -structure  $A$  be written on the input tape of  $T^{\text{MOD}[\Upsilon_{\tau_{<}}]}$ , and  $n$  denote  $\|A\|$ . At first, the machine  $T^{\text{MOD}[\Upsilon_{\tau_{<}}]}$  computes  $n$  as  $|\langle A \rangle|$  since the equality  $\|A\| = |\langle A \rangle|$  holds in case of the vocabulary  $\sigma_{<}$ . Then, the machine  $T^{\text{MOD}[\Upsilon_{\tau_{<}}]}$  rewrites the input string  $\langle A \rangle$  onto the query tape. Next, the machine  $T^{\text{MOD}[\Upsilon_{\tau_{<}}]}$  will transform the  $\sigma_{<}$ -structure  $A$  into a  $\tau_{<}$ -structure  $B$  such that  $\|B\| = \|A\|$ , either  $R_1^B = \{0\}^{a_1-1} \times R^A$  if  $a_1 > 1$ , or  $R_1^B = R^A$  if  $a_1 = 1$ ;  $R_2^B = \emptyset, \dots, R_m^B = \emptyset$ . For this purpose,  $T^{\text{MOD}[\Upsilon_{\tau_{<}}]}$  appends  $N$  zeros to  $\langle A \rangle$  on the query tape, obtaining the string  $w = \langle A \rangle 0^N$ , where  $N = n^{a_1} + n^{a_2} + \dots + n^{a_m} - n$ . Note that  $w$  encodes the  $\tau_{<}$ -structure  $B$  since the initial substring  $\langle A \rangle 0^{n^{a_1}-n}$  encodes the relation  $R_1^B$ , and (if any) the other substrings  $0^{n^{a_2}}, \dots, 0^{n^{a_m}}$  encode the relations  $R_2^B, \dots, R_m^B$  respectively. Then,  $T^{\text{MOD}[\Upsilon_{\tau_{<}}]}$  enters the query state *QUE*. If  $B \models \Upsilon_{\tau_{<}}$ , then  $T^{\text{MOD}[\Upsilon_{\tau_{<}}]}$  goes into state *YES*, otherwise it enters state *NO*. At the next step,  $T^{\text{MOD}[\Upsilon_{\tau_{<}}]}$  moves to state *ACC* from state *YES* (i.e.  $T^{\text{MOD}[\Upsilon_{\tau_{<}}]}$  accepts  $A$ ). Otherwise,  $T^M$  rejects  $A$ .

It is obvious that  $T^{\text{MOD}[\Upsilon_{\tau_{<}}]}$  realizes a logspace reduction from  $\text{MOD}[\Upsilon(R(x))]$  to  $\text{MOD}[\Upsilon_{\tau_{<}}]$  since  $A$  is a model of  $\Upsilon(R(x))$  if and only if  $B$  is a model of  $\Upsilon_{\tau_{<}}$ . Therefore,  $\text{MOD}[\Upsilon_{\tau_{<}}]$  is  $\mathcal{N}$ -complete as well as  $\text{MOD}[\Upsilon(R(x))]$ . Thus, the  $\mathcal{L}(\tau_{<})$ -sentence  $\Upsilon_{\tau_{<}}$  indeed defines some  $\mathcal{N}$ -complete problem.

With each  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentence  $\Gamma$  and with each  $\mathcal{N}$ -specific Turing machine  $T^{\text{MOD}[\Gamma]}$ , we associate the following set of  $\tau_{<}$ -structures:

$$S_{\Gamma, T^{\text{MOD}[\Gamma]}}^< \triangleq \{ A \in \text{STRUC}[\tau_{<}] \mid (\forall B \in \text{STRUC}[\tau_{<}]) [ \|B\| > \ell^{(3)}(|\langle A \rangle|) \vee (T^{\text{MOD}[\Gamma]} \text{ accepts } \langle B \rangle \Leftrightarrow B \models \Upsilon_{\tau_{<}}) ] \}.$$

The set  $S_{\Gamma, T^{\text{MOD}[\Gamma]}}^<$  has some interesting properties. Note that if the condition “ $T^{\text{MOD}[\Gamma]}$  accepts  $\langle B \rangle \Leftrightarrow B \models \Upsilon_{\tau_{<}}$ ” is satisfied for all  $B \in \text{STRUC}[\tau_{<}]$ , then  $T^{\text{MOD}[\Gamma]}$  is a Turing reduction from  $\text{MOD}[\Upsilon_{\tau_{<}}]$  to  $\text{MOD}[\Gamma]$ . In this case,  $\text{MOD}[\Gamma]$  is a  $\mathcal{N}$ -complete model class, and  $S_{\Gamma, T^{\text{MOD}[\Gamma]}}^< = \text{STRUC}[\tau_{<}]$ . If this is not the case, then  $S_{\Gamma, T^{\text{MOD}[\Gamma]}}^<$  is finite. Let us show how to find a  $\text{FO}(\tau_{<})$ -sentence that defines the model class  $S_{\Gamma, T^{\text{MOD}[\Gamma]}}^<$ .

The complexity class  $\text{DTIME}[\log n]$  consists of all problems decidable in logarithmic time in the input length  $n$ . For logarithmic time, an appropriate model of computation is random access machines which can directly access any memory cell by means of indices. We have the following theorem.

**Theorem 3**  $S_{\Gamma, T^{\text{MOD}[\Gamma]}}^< \in \text{DTIME}[\log n]$ .

**PROOF.** Let us use a random access machine to decide whether  $A \in S_{\Gamma, T^{\text{MOD}[\Gamma]}}^<$  or not, given a  $\tau_{<}$ -structure  $A$ . As stated in [11], in complexity theory,  $n$  usually denotes the size of the input. However, in finite model theory,  $n$  denotes the cardinality of the universe. In order to clear up confusion, let  $\hat{n}$  denote  $|\langle A \rangle|$ .

First, we need to compute the number  $\ell^{(3)}(\hat{n})$ . For this purpose, we compute  $\hat{n}$ , using a binary search [1]. Since we use a random access machine, this can be done in time  $O(\log \hat{n})$ . Then, we count the number  $\ell(\hat{n})$  of bits in  $\hat{n}$  represented in binary. Likewise, we count the number  $\ell^{(2)}(\hat{n})$  of bits in  $\ell(\hat{n})$  and then the number  $\ell^{(3)}(\hat{n})$  of bits in  $\ell^{(2)}(\hat{n})$ . It is obvious that the total time necessary for computing  $\ell^{(3)}(\hat{n})$  is  $O(\log \hat{n})$ .

Second, we enumerate all  $\tau_{<}$ -structures  $B \in \text{STRUC}[\tau_{<}]$  such that  $\|B\| \leq \ell^{(3)}(\hat{n})$ . Since  $|\langle B \rangle| \leq p(\|B\|)$ , the value  $|\langle B \rangle|$  is bounded above by  $p(\log \log \log \hat{n})$ , where  $p$  is a polynomial dependent on the vocabulary  $\tau_{<}$ . Therefore, this enumeration takes time  $O(2^{p(\log \log \log \hat{n})})$ . Then, for each  $\tau_{<}$ -structure  $B$  of the enumeration, we need to verify the following condition:

$$T^{\text{MOD}[\Gamma]} \text{ accepts } \langle B \rangle \Leftrightarrow B \models \Upsilon_{\tau_{<}}. \quad (1)$$

If condition (1) is not satisfied for some  $\tau_{<}$ -structure  $B$  of the enumeration, then the input  $\tau_{<}$ -structure  $A$  does not belong to  $S_{\Gamma, T^{\text{MOD}[\Gamma]}}^<$ . Otherwise,  $A \in S_{\Gamma, T^{\text{MOD}[\Gamma]}}^<$ . Now, let us estimate the running time for verifying condition (1).

We can decide whether “ $B \models \Upsilon_{\tau_{<}}$ ” for one  $\tau_{<}$ -structure  $B$  in time  $O(2^{p_1(\log \log \log \hat{n})})$ , where  $p_1$  is a polynomial. This follows from the fact that any model-checking problem in  $\mathcal{N}$  is decidable at most in exponential time. In a similar manner, the verification of “ $T^{\text{MOD}[\Gamma]}$  accepts  $\langle B \rangle$ ” takes time  $O(2^{p_2(\log \log \log \hat{n})})$ , where  $p_2$  is a polynomial. This follows from the fact that a simulation of running  $T^{\text{MOD}[\Gamma]}$  together with a simulation of oracle queries to  $\text{MOD}[\Gamma]$  requires at most exponential time as well. Therefore, the verification of (1) takes time  $O(2^{p_1(\log \log \log \hat{n})} + 2^{p_2(\log \log \log \hat{n})})$ , or, in short,  $O(2^{p'(\log \log \log \hat{n})})$ , where either  $p' = p_1$  in case of  $\lim_{\hat{n} \rightarrow \infty} \frac{p_2(\hat{n})}{p_1(\hat{n})} < \infty$ , or  $p' = p_2$  otherwise.

Consequently, the total time to verify (1) for all  $\tau_{<}$ -structures  $B$  of the enumeration is  $O(2^{p(\log \log \log \hat{n})} \cdot 2^{p'(\log \log \log \hat{n})})$ , or, in short,  $O(2^{p''(\log \log \log \hat{n})})$ , where  $p'' = p + p'$ . Note that  $\lim_{\hat{n} \rightarrow \infty} \frac{2^{p''(\log \log \log \hat{n})}}{\log \hat{n}} = 0$  for any polynomial  $p''$ . Then, this time can be approximately estimated at most as  $O(\log \hat{n})$ .

Thus, the overall running time consists of the time for computing  $\ell^{(3)}(\hat{n})$  and of the time for verifying (1) on all  $\tau_{<}$ -structures  $B$  of the enumeration. These times are both estimated as  $O(\log \hat{n})$ . Therefore, so is the overall running time. This concludes the proof of the theorem.  $\square$

**Corollary 4** *Given a  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentence  $\Gamma$  and a  $\mathcal{N}$ -specific Turing machine  $T^{\text{MOD}[\Gamma]}$ , one can construct a  $\text{FO}(\tau_{<})$ -sentence that defines  $S_{\Gamma, T^{\text{MOD}[\Gamma]}}^<$ .*

**PROOF.** By Theorem 3, we can use a random access machine  $M$  to recognize  $S_{\Gamma, T^{\text{MOD}[\Gamma]}}^<$ , where  $M$  runs in time  $O(\log n)$ . Then, one can construct the  $\text{FO}(\tau_{<})$ -sentence from  $M$  as shown in [11], provided that first order logic is extended with the numerical predicate BIT.  $\square$

Then, by  $\gamma_{\Gamma, T^{\text{MOD}}[\Gamma]}$  we denote the constructed  $\text{FO}(\tau_{<})$ -sentence defining the model class  $S_{\Gamma, T^{\text{MOD}}[\Gamma]}^{\leq}$ . This  $\gamma_{\Gamma, T^{\text{MOD}}[\Gamma]}$  characterizes the corresponding Turing machine  $T^{\text{MOD}}[\Gamma]$  in two different ways. First,  $T^{\text{MOD}}[\Gamma]$  is a Turing reduction from  $\text{MOD}[\Upsilon_{\tau_{<}}]$  to  $\text{MOD}[\Gamma]$  if and only if  $\gamma_{\Gamma, T^{\text{MOD}}[\Gamma]}$  is logically valid. Second,  $T^{\text{MOD}}[\Gamma]$  does not realize any Turing reduction from  $\text{MOD}[\Upsilon_{\tau_{<}}]$  to  $\text{MOD}[\Gamma]$  if and only if  $\text{MOD}[\gamma_{\Gamma, T^{\text{MOD}}[\Gamma]}]$  is finite. We call the  $\text{FO}(\tau_{<})$ -sentence  $\gamma_{\Gamma, T^{\text{MOD}}[\Gamma]}$  the characteristic sentence for the pair  $(\Gamma, T^{\text{MOD}}[\Gamma])$ . The following theorem holds.

**Theorem 5** *Let  $\mathcal{L}_{\mathcal{N}}$  be a logic capturing a complexity class  $\mathcal{N}$  on  $\text{STRUC}[\tau_{<}]$ , and  $\Pi \subseteq \text{STRUC}[\tau_{<}]$  a model class. Then,  $\Pi$  is  $\mathcal{N}$ -complete if and only if there exists a  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentence  $\Lambda$  such that  $\text{MOD}[\Lambda] = \Pi$  and  $\Lambda$  is of the form*

$$(\gamma_{\Gamma, T^{\text{MOD}}[\Gamma]} \wedge \Gamma) \vee (\neg \gamma_{\Gamma, T^{\text{MOD}}[\Gamma]} \wedge \Upsilon_{\tau_{<}}) \quad (2)$$

where  $\Gamma$  is a  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentence;  $T^{\text{MOD}}[\Gamma]$  a  $\mathcal{N}$ -specific Turing machine;  $\gamma_{\Gamma, T^{\text{MOD}}[\Gamma]}$  the characteristic sentence for  $(\Gamma, T^{\text{MOD}}[\Gamma])$ .

**PROOF.** For short, we denote the form (2) by  $\Phi_{\Gamma, T^{\text{MOD}}[\Gamma]}$ .

First, we will prove that  $\Phi_{\Gamma, T^{\text{MOD}}[\Gamma]}$  defines a  $\mathcal{N}$ -complete problem for any pair  $(\Gamma, T^{\text{MOD}}[\Gamma])$ . Let  $T^{\text{MOD}}[\Gamma]$  be a Turing reduction from  $\text{MOD}[\Upsilon_{\tau_{<}}]$  to  $\text{MOD}[\Gamma]$ . Then,  $\gamma_{\Gamma, T^{\text{MOD}}[\Gamma]}$  is logically valid. In this case, the  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentence  $\Phi_{\Gamma, T^{\text{MOD}}[\Gamma]}$  is logically equivalent to  $\Gamma$ , and  $\Gamma$  defines a  $\mathcal{N}$ -complete problem. Therefore,  $\Phi_{\Gamma, T^{\text{MOD}}[\Gamma]}$  defines the same  $\mathcal{N}$ -complete problem as well. Now, let  $T^{\text{MOD}}[\Gamma]$  be not a Turing reduction from  $\text{MOD}[\Upsilon_{\tau_{<}}]$  to  $\text{MOD}[\Gamma]$ . Then,  $\text{MOD}[\gamma_{\Gamma, T^{\text{MOD}}[\Gamma]}]$  is finite, and  $\text{MOD}[\neg \gamma_{\Gamma, T^{\text{MOD}}[\Gamma]}]$  is cofinite. Therefore, the model class  $\text{MOD}[\Phi_{\Gamma, T^{\text{MOD}}[\Gamma]}]$  differs from the model class  $\text{MOD}[\Upsilon_{\tau_{<}}]$  only in a finite set of  $\tau_{<}$ -structures, i.e.  $\text{MOD}[\Phi_{\Gamma, T^{\text{MOD}}[\Gamma]}] \setminus \text{MOD}[\Upsilon_{\tau_{<}}]$  and  $\text{MOD}[\Upsilon_{\tau_{<}}] \setminus \text{MOD}[\Phi_{\Gamma, T^{\text{MOD}}[\Gamma]}]$  are both finite. Hence,  $\text{MOD}[\Phi_{\Gamma, T^{\text{MOD}}[\Gamma]}]$  is  $\mathcal{N}$ -complete. Consequently,  $\Phi_{\Gamma, T^{\text{MOD}}[\Gamma]}$  defines a  $\mathcal{N}$ -complete problem in this case as well.

Second, we will prove that any  $\mathcal{N}$ -complete problem can be represented by means of the form  $\Phi_{\Gamma, T^{\text{MOD}}[\Gamma]}$ . Let  $\Gamma$  be a  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentence defining an arbitrary  $\mathcal{N}$ -complete problem, and  $T^{\text{MOD}}[\Gamma]$  a  $\mathcal{N}$ -specific Turing machine realizing a Turing reduction from  $\text{MOD}[\Upsilon_{\tau_{<}}]$  to  $\text{MOD}[\Gamma]$ . Then,  $\gamma_{\Gamma, T^{\text{MOD}}[\Gamma]}$  is logically valid. In this case, the  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentence  $\Gamma$  is logically equivalent to the  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentence  $\Phi_{\Gamma, T^{\text{MOD}}[\Gamma]}$ . Since  $\Gamma$  defines a  $\mathcal{N}$ -complete problem, so does  $\Phi_{\Gamma, T^{\text{MOD}}[\Gamma]}$ . This concludes the proof of the theorem.  $\square$

Thus, the form (2) can serve as a canonical form providing a syntactic tool for showing  $\mathcal{N}$ -completeness: if a problem on ordered structures in  $\mathcal{N}$  is defined by a sentence of the form, then the problem proves to be  $\mathcal{N}$ -complete via Turing reductions.

## 4.2 Canonical form for complete problems on unordered structures

We will introduce a canonical form for  $\mathcal{N}$ -complete problems on unordered non-Aristotelian structures. We denote by  $\sigma$  the vocabulary  $\{R^2\}$ . In this section, we by  $\mathcal{N}$  mean one of the following complexity classes:  $\text{coNP}$ ,  $\text{NP}$ , or  $\text{PSPACE}$ . Let  $\mathcal{L}_{\mathcal{N}}$  be a logic capturing  $\mathcal{N}$  on  $\text{STRUC}[\sigma]$ . For definiteness, we assume that  $\mathcal{L}_{\text{coNP}}$ ,  $\mathcal{L}_{\text{NP}}$ , and  $\mathcal{L}_{\text{PSPACE}}$  stand for the following logics:  $\text{SO}\forall$ ,  $\text{SO}\exists$ , and  $\text{SO}(\text{PFP})$  respectively. Let  $\Upsilon(R(x, y))$  be a distinguished  $\mathcal{L}(\sigma)$ -sentence defining some  $\mathcal{N}$ -complete model class.

We need to be able to construct sentences defining  $\mathcal{N}$ -complete problems for various non-Aristotelian vocabularies, using an operator over  $\Upsilon(R(x, y))$ . Let  $\tau = \{R_1^{a_1}, \dots, R_m^{a_m}\}$  be a fixed arbitrary non-Aristotelian vocabulary. We define the operator  $T_\tau$  for mapping  $\Upsilon(R(x, y))$  to a  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentence as follows. We find the least number  $k$  which is subject to  $1 \leq k \leq m$  and  $a_k > 1$ . If  $a_k = 2$ , then  $T_\tau(\Upsilon(R(x, y))) \triangleq \Upsilon[R(x, y)/R_k(x, y)]$ . If  $a_k > 2$ , then  $T_\tau(\Upsilon(R(x, y))) \triangleq \Upsilon[R(x, y)/\exists z R_k(x, y, \underbrace{z, \dots, z}_{a_k - 2 \text{ times}})]$ , where  $z$  is a new variable (we assume that  $z$  does not occur anywhere in  $\Upsilon(R(x, y))$ ). For short, let  $\Upsilon_\tau$  denote  $T_\tau(\Upsilon(R(x, y)))$ .

Let us show that the  $\mathcal{L}(\tau)$ -sentence  $\Upsilon_\tau$  defines a  $\mathcal{N}$ -complete problem. We can easily reduce  $\text{MOD}[\Upsilon(R(x, y))]$  to  $\text{MOD}[\Upsilon_\tau]$  in the following way. Let  $A$  be an arbitrary  $\sigma$ -structure. Then, we take a  $\tau$ -structure  $B$  such that  $\|B\| = \|A\|$ , either  $R_k^B = R^A$  if  $a_k = 2$ , or  $R_k^B = R^A \times \{0\}^{a_k - 2}$  if  $a_k > 2$ ;  $R_1^B = \emptyset, \dots, R_{k-1}^B = \emptyset, R_{k+1}^B = \emptyset, \dots, R_m^B = \emptyset$ . Note that  $A$  is a model of  $\Upsilon(R(x, y))$  if and only if  $B$  is a model of  $\Upsilon_\tau$ . It is obvious that  $\text{MOD}[\Upsilon_\tau]$  is  $\mathcal{N}$ -complete as well as  $\text{MOD}[\Upsilon(R(x, y))]$ . Thus, the  $\mathcal{L}(\tau)$ -sentence  $\Upsilon_\tau$  indeed defines some  $\mathcal{N}$ -complete problem.

With each  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentence  $\Gamma$  and with each  $\mathcal{N}$ -specific Turing machine  $T^{\text{MOD}[\Gamma]}$ , we associate the following set of  $\tau$ -structures:

$$S_{\Gamma, T^{\text{MOD}[\Gamma]}} \triangleq \{ A \in \text{STRUC}[\tau] \mid (\forall B \in \text{STRUC}[\tau]) [ \|B\| > \ell^{(2)}(\|A\|) \vee (T^{\text{MOD}[\Gamma]} \text{ accepts } \langle B \rangle \Leftrightarrow B \models \Upsilon_\tau) ] \}.$$

Like  $S_{\Gamma, T^{\text{MOD}[\Gamma]}}^<$ , the set  $S_{\Gamma, T^{\text{MOD}[\Gamma]}}$  has similar properties. Note that if the condition “ $T^{\text{MOD}[\Gamma]}$  accepts  $\langle B \rangle \Leftrightarrow B \models \Upsilon_\tau$ ” is satisfied for all  $B \in \text{STRUC}[\tau]$ , then  $T^{\text{MOD}[\Gamma]}$  is a Turing reduction from  $\text{MOD}[\Upsilon_\tau]$  to  $\text{MOD}[\Gamma]$ . In this case,  $\text{MOD}[\Gamma]$  is a  $\mathcal{N}$ -complete model class, and  $S_{\Gamma, T^{\text{MOD}[\Gamma]}} = \text{STRUC}[\tau]$ . If this is not the case, then  $S_{\Gamma, T^{\text{MOD}[\Gamma]}}$  is finite. Let us show how to find a  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentence that defines the model class  $S_{\Gamma, T^{\text{MOD}[\Gamma]}}$ .

The complexity class  $\text{DTIME}[n]$  consists of all problems decidable in linear time in the input size  $n$ . The following theorem holds.

**Theorem 6**  $S_{\Gamma, T^{\text{MOD}[\Gamma]}} \in \text{DTIME}[n]$ .

**PROOF.** It is similar to the proof of Theorem 3 in many respects. Let us decide in linear time whether  $A \in S_{\Gamma, T^{\text{MOD}[\Gamma]}}$  or not, given a  $\tau$ -structure  $A$ . Let  $\hat{n}$  denote  $\|A\|$ .

First, we need to compute the number  $\ell^{(2)}(\hat{n})$ . Suppose that we use a usual Turing machine rather than a random access machine. In this case, it is obvious that the time necessary for computing  $\ell^{(2)}(\hat{n})$  is  $O(\hat{n})$ .

Second, we enumerate all  $\tau$ -structures  $B \in \text{STRUC}[\tau]$  such that  $\|B\| \leq \ell^{(2)}(\hat{n})$ . Since  $|\langle B \rangle| \leq p(\|B\|)$ , the value  $|\langle B \rangle|$  is bounded above by  $p(\log \log \hat{n})$ , where  $p$  is a polynomial dependent on the vocabulary  $\tau$ . Therefore, this enumeration takes time  $O(2^{p(\log \log \hat{n})})$ . Then, for each  $\tau$ -structure  $B$  of the enumeration, we need to verify the following condition:

$$T^{\text{MOD}[\Gamma]} \text{ accepts } \langle B \rangle \Leftrightarrow B \models \Upsilon_\tau. \quad (3)$$

If condition (3) is not satisfied for some  $\tau$ -structure  $B$  of the enumeration, then the input  $\tau$ -structure  $A$  does not belong to  $S_{\Gamma, T^{\text{MOD}[\Gamma]}}$ . Otherwise,  $A \in S_{\Gamma, T^{\text{MOD}[\Gamma]}}$ . Now, let us estimate the running time for verifying condition (3).

We can decide whether “ $B \models \Upsilon_\tau$ ” for one  $\tau$ -structure  $B$  in time  $O(2^{p_1(\log \log \hat{n})})$ , where  $p_1$  is a polynomial. This follows from the fact that any model-checking problem in  $\mathcal{N}$  is decidable at most in exponential time. In a similar manner, the verification of “ $T^{\text{MOD}[\Gamma]} \text{ accepts } \langle B \rangle$ ” takes time  $O(2^{p_2(\log \log \hat{n})})$ , where  $p_2$  is a polynomial. This follows from the fact that a simulation of running  $T^{\text{MOD}[\Gamma]}$  together with a simulation of oracle queries to  $\text{MOD}[\Gamma]$  requires at most exponential time as well. Therefore, the verification of (3) takes time  $O(2^{p_1(\log \log \hat{n})} + 2^{p_2(\log \log \hat{n})})$ , or, in short,  $O(2^{p'(\log \log \hat{n})})$ , where either  $p' = p_1$  in case of  $\lim_{\hat{n} \rightarrow \infty} \frac{p_2(\hat{n})}{p_1(\hat{n})} < \infty$ , or  $p' = p_2$  otherwise.

Consequently, the total time to verify (3) for all  $\tau$ -structures  $B$  of the enumeration is  $O(2^{p(\log \log \hat{n})} \cdot 2^{p'(\log \log \hat{n})})$ , or, in short,  $O(2^{p''(\log \log \hat{n})})$ , where  $p'' = p + p'$ . Note that  $\lim_{\hat{n} \rightarrow \infty} \frac{2^{p''(\log \log \hat{n})}}{\hat{n}} = 0$  for any polynomial  $p''$ . Then, this time can be approximately estimated at most as  $O(\hat{n})$ .

Thus, the overall running time consists of the time for computing  $\ell^{(2)}(\hat{n})$  and of the time for verifying (3) on all  $\tau$ -structures  $B$  of the enumeration. The both of these times are estimated as  $O(\hat{n})$ . Therefore, so is the overall running time. This concludes the proof of the theorem.  $\square$

**Corollary 7** *Given a  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentence  $\Gamma$  and a  $\mathcal{N}$ -specific Turing machine  $T^{\text{MOD}[\Gamma]}$ , one can construct a  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentence that defines  $S_{\Gamma, T^{\text{MOD}[\Gamma]}}$ .*

**PROOF.** By Fagin’s theorem [6], in case of  $\mathcal{N} = \text{NP}$ , we can construct a  $\text{SO}\exists(\tau)$ -sentence to define the model class  $S_{\Gamma, T^{\text{MOD}[\Gamma]}}$  since  $S_{\Gamma, T^{\text{MOD}[\Gamma]}} \in \text{DTIME}[n]$ . In case of  $\mathcal{N} = \text{PSPACE}$ , we can use the same  $\text{SO}\exists(\tau)$ -sentence since  $\text{SO}\exists$  is a fragment of  $\text{SO}(\text{PFP})$ . In case of  $\mathcal{N} = \text{coNP}$ , we can at first construct a  $\text{SO}\exists(\tau)$ -sentence to define the model class  $\text{STRUC}[\tau] \setminus S_{\Gamma, T^{\text{MOD}[\Gamma]}}$  since  $\text{STRUC}[\tau] \setminus S_{\Gamma, T^{\text{MOD}[\Gamma]}} \in \text{DTIME}[n]$  as well as  $S_{\Gamma, T^{\text{MOD}[\Gamma]}} \in \text{DTIME}[n]$ . Then, the  $\text{SO}\forall(\tau)$ -sentence defining  $S_{\Gamma, T^{\text{MOD}[\Gamma]}}$  is obtained by logical negation of this  $\text{SO}\exists(\tau)$ -sentence.  $\square$

Then, by  $\Theta_{\Gamma, T^{\text{MOD}[\Gamma]}}$  we denote the constructed  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentence defining the model class  $S_{\Gamma, T^{\text{MOD}[\Gamma]}}$ . This  $\Theta_{\Gamma, T^{\text{MOD}[\Gamma]}}$  characterizes the corresponding Turing

machine  $T^{\text{MOD}[\Gamma]}$  in two different ways. First,  $T^{\text{MOD}[\Gamma]}$  is a Turing reduction from  $\text{MOD}[\Upsilon_\tau]$  to  $\text{MOD}[\Gamma]$  if and only if  $\Theta_{\Gamma, T^{\text{MOD}[\Gamma]}}$  is logically valid. Second,  $T^{\text{MOD}[\Gamma]}$  does not realize any Turing reduction from  $\text{MOD}[\Upsilon_\tau]$  to  $\text{MOD}[\Gamma]$  if and only if  $\text{MOD}[\Theta_{\Gamma, T^{\text{MOD}[\Gamma]}}]$  is finite. We call the  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentence  $\Theta_{\Gamma, T^{\text{MOD}[\Gamma]}}$  the characteristic sentence for the pair  $(\Gamma, T^{\text{MOD}[\Gamma]})$ .

Similarly, by  $\overline{\Theta}_{\Gamma, T^{\text{MOD}[\Gamma]}}$  we denote a  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentence that defines the model class  $\text{STRUC}[\tau] \setminus S_{\Gamma, T^{\text{MOD}[\Gamma]}}$ . Given a  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentence  $\Gamma$  and a  $\mathcal{N}$ -specific Turing machine  $T^{\text{MOD}[\Gamma]}$ , this  $\overline{\Theta}_{\Gamma, T^{\text{MOD}[\Gamma]}}$  can be also constructed since  $\text{STRUC}[\tau] \setminus S_{\Gamma, T^{\text{MOD}[\Gamma]}} \in \text{DTIME}[n]$  as well as  $S_{\Gamma, T^{\text{MOD}[\Gamma]}} \in \text{DTIME}[n]$ . Note that  $\overline{\Theta}_{\Gamma, T^{\text{MOD}[\Gamma]}}$  is logically equivalent to  $\neg\Theta_{\Gamma, T^{\text{MOD}[\Gamma]}}$ . However, we cannot directly use  $\neg\Theta_{\Gamma, T^{\text{MOD}[\Gamma]}}$  to construct a canonical form since  $\neg\Theta_{\Gamma, T^{\text{MOD}[\Gamma]}}$  is not a  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentence in case of  $\mathcal{N} \in \{\text{NP}, \text{coNP}\}$ . Therefore, we will use  $\overline{\Theta}_{\Gamma, T^{\text{MOD}[\Gamma]}}$  instead. We call the  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentence  $\overline{\Theta}_{\Gamma, T^{\text{MOD}[\Gamma]}}$  the complementary sentence for  $\Theta_{\Gamma, T^{\text{MOD}[\Gamma]}}$ . Then, the following theorem holds.

**Theorem 8** *Let  $\mathcal{L}_{\mathcal{N}}$  be a logic capturing a complexity class  $\mathcal{N}$  on  $\text{STRUC}[\tau]$ , and  $\Pi \subseteq \text{STRUC}[\tau]$  a model class. Then,  $\Pi$  is  $\mathcal{N}$ -complete if and only if there exists a  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentence  $\Lambda$  such that  $\text{MOD}[\Lambda] = \Pi$  and  $\Lambda$  is of the form*

$$(\Theta_{\Gamma, T^{\text{MOD}[\Gamma]}} \wedge \Gamma) \vee (\overline{\Theta}_{\Gamma, T^{\text{MOD}[\Gamma]}} \wedge \Upsilon_\tau) \quad (4)$$

where  $\Gamma$  is a  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentence;  $T^{\text{MOD}[\Gamma]}$  a  $\mathcal{N}$ -specific Turing machine;  $\Theta_{\Gamma, T^{\text{MOD}[\Gamma]}}$  the characteristic sentence for the pair  $(\Gamma, T^{\text{MOD}[\Gamma]})$ ;  $\overline{\Theta}_{\Gamma, T^{\text{MOD}[\Gamma]}}$  the complementary sentence for  $\Theta_{\Gamma, T^{\text{MOD}[\Gamma]}}$ .

**PROOF.** For short, we denote the form (4) by  $\Phi_{\Gamma, T^{\text{MOD}[\Gamma]}}$ .

First, we will prove that  $\Phi_{\Gamma, T^{\text{MOD}[\Gamma]}}$  defines a  $\mathcal{N}$ -complete problem for any pair  $(\Gamma, T^{\text{MOD}[\Gamma]})$ . Let  $T^{\text{MOD}[\Gamma]}$  be a Turing reduction from  $\text{MOD}[\Upsilon_\tau]$  to  $\text{MOD}[\Gamma]$ . Then,  $\Theta_{\Gamma, T^{\text{MOD}[\Gamma]}}$  is logically valid. In this case, the  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentence  $\Phi_{\Gamma, T^{\text{MOD}[\Gamma]}}$  is logically equivalent to  $\Gamma$ , and  $\Gamma$  defines a  $\mathcal{N}$ -complete problem. Therefore,  $\Phi_{\Gamma, T^{\text{MOD}[\Gamma]}}$  defines the same  $\mathcal{N}$ -complete problem as well. Now, let  $T^{\text{MOD}[\Gamma]}$  be not a Turing reduction from  $\text{MOD}[\Upsilon_\tau]$  to  $\text{MOD}[\Gamma]$ . Then,  $\text{MOD}[\Theta_{\Gamma, T^{\text{MOD}[\Gamma]}}]$  is finite, and  $\text{MOD}[\overline{\Theta}_{\Gamma, T^{\text{MOD}[\Gamma]}}]$  is cofinite. Therefore, the model class  $\text{MOD}[\Phi_{\Gamma, T^{\text{MOD}[\Gamma]}}]$  differs from the model class  $\text{MOD}[\Upsilon_\tau]$  only in a finite set of  $\tau$ -structures, i.e.  $\text{MOD}[\Phi_{\Gamma, T^{\text{MOD}[\Gamma]}}] \setminus \text{MOD}[\Upsilon_\tau]$  and  $\text{MOD}[\Upsilon_\tau] \setminus \text{MOD}[\Phi_{\Gamma, T^{\text{MOD}[\Gamma]}}]$  are both finite. Hence,  $\text{MOD}[\Phi_{\Gamma, T^{\text{MOD}[\Gamma]}}]$  is  $\mathcal{N}$ -complete. Consequently,  $\Phi_{\Gamma, T^{\text{MOD}[\Gamma]}}$  defines a  $\mathcal{N}$ -complete problem in this case as well.

Second, we will prove that any  $\mathcal{N}$ -complete problem can be represented by means of the form  $\Phi_{\Gamma, T^{\text{MOD}[\Gamma]}}$ . Let  $\Gamma$  be a  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentence defining an arbitrary  $\mathcal{N}$ -complete problem, and  $T^{\text{MOD}[\Gamma]}$  a  $\mathcal{N}$ -specific Turing machine realizing a Turing reduction from  $\text{MOD}[\Upsilon_\tau]$  to  $\text{MOD}[\Gamma]$ . Then,  $\Theta_{\Gamma, T^{\text{MOD}[\Gamma]}}$  is logically valid. In this case, the  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentence  $\Gamma$  is logically equivalent to the  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentence  $\Phi_{\Gamma, T^{\text{MOD}[\Gamma]}}$ . Since  $\Gamma$  defines a  $\mathcal{N}$ -complete problem, so does  $\Phi_{\Gamma, T^{\text{MOD}[\Gamma]}}$ . This concludes the proof of the theorem.  $\square$

Thus, the form (4) can serve as a canonical form providing a syntactic tool for showing  $\mathcal{N}$ -completeness: if a problem on unordered structures in  $\mathcal{N}$  is defined by a sentence of the form, then the problem proves to be  $\mathcal{N}$ -complete via Turing reductions.

## 5 Logics for completeness

At first, we will consider ordered structures. Let an arbitrary vocabulary  $\tau_{<}$  containing  $<$  be given. Let us address the following question. Can one recursively enumerate all  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentences that define  $\mathcal{N}$ -complete problems on  $\text{STRUC}[\tau_{<}]$ , where  $\mathcal{N}$  denotes one of the following complexity classes: NL, P, coNP, NP, or PSPACE? We will answer this question in the negative.

We will use notions of context-free languages [7], and exploit them in a similar manner as in [15]. Let  $\Sigma$  be a fixed alphabet containing at least two symbols. By  $G$  and  $L(G)$  we mean a context-free grammar and the context-free language defined by this grammar respectively. We assume that there cannot be any finite  $\mathcal{N}$ -complete set. Suppose that  $\Phi_1, \Phi_2, \dots$  is a recursive enumeration of all  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentences defining  $\mathcal{N}$ -complete problems. With every context-free grammar  $G$  with the terminal alphabet  $\Sigma$ , we associate the following set:

$$S_G^{\leq} \triangleq \{ A \in \text{STRUC}[\tau_{<}] \mid (\forall w \in \Sigma^*)[|w| > \ell^{(3)}(|\langle A \rangle|) \vee w \in L(G)] \}.$$

Since the decision problem of  $w \in? L(G)$  requires at most polynomial time, we have  $S_G^{\leq} \in \text{DTIME}[\log n]$  by analogy with Theorem 3. Then, there is a  $\text{FO}(\tau_{<})$ -sentence  $\gamma_G$  that defines  $S_G^{\leq}$ . Note that if  $L(G) = \Sigma^*$ , then  $\gamma_G$  is logically valid. Otherwise, the set  $\text{MOD}[\gamma_G]$  is finite. It follows that the  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentence  $\gamma_G \wedge \Upsilon_{\tau_{<}}$  defines a  $\mathcal{N}$ -complete problem if and only if the statement  $L(G) = \Sigma^*$  holds.

Let an arbitrary context-free grammar  $G$  with the terminal alphabet  $\Sigma$  be given. Let us decide whether  $L(G) = \Sigma^*$  or not. We concurrently start the following two algorithms. First, we enumerate all strings in  $\Sigma^*$ . If we can find a string  $w$  such that  $w \notin L(G)$ , then  $L(G) \neq \Sigma^*$ . Second, we construct the  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentence  $\gamma_G \wedge \Upsilon_{\tau_{<}}$  for this grammar  $G$ . Then, we enumerate the  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentences  $\Phi_1, \Phi_2, \dots$ . If we can find  $\Phi_i$  such that  $\Phi_i = \gamma_G \wedge \Upsilon_{\tau_{<}}$ , then  $L(G) = \Sigma^*$ . Hence, we can algorithmically decide whether  $L(G) = \Sigma^*$  or not. However, this contradicts the fact that the decision problem of  $L(G) =? \Sigma^*$  is algorithmically undecidable [7]. Thus, one cannot recursively enumerate all  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentences that define  $\mathcal{N}$ -complete problems on  $\text{STRUC}[\tau_{<}]$ .

Nevertheless, we can use form (2) in order to recursively enumerate (not all)  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentences that define all  $\mathcal{N}$ -complete problems. Let  $\mathcal{C}_{\mathcal{N}}(\tau_{<})$  denote the following set of all  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentences of form (2):

$$\{ \Phi \in \mathcal{L}_{\mathcal{N}}(\tau_{<}) \mid \Phi = (\gamma_{\Gamma, T^{\text{MOD}}[\Gamma]} \wedge \Gamma) \vee (\neg \gamma_{\Gamma, T^{\text{MOD}}[\Gamma]} \wedge \Upsilon_{\tau_{<}}); \Gamma \in \mathcal{L}_{\mathcal{N}}(\tau_{<}), \\ T^{\text{MOD}}[\Gamma] \text{ is a } \mathcal{N}\text{-specific Turing machine} \}.$$

Note that  $\mathcal{C}_{\mathcal{N}}(\tau_{<})$  is recursively enumerable since we can effectively enumerate all possible pairs  $(\Gamma, T^{\text{MOD}}[\Gamma])$ . Therefore, the set of all  $\mathcal{N}$ -complete prob-

lems on  $\text{STRUC}[\tau_<]$ , defined by sentences in  $\mathcal{C}_{\mathcal{N}}(\tau_<)$ , is recursively enumerable as well.

However, it is hard to determine whether  $\mathcal{C}_{\mathcal{N}}(\tau_<)$  is recursive. In order to be able to decide whether  $\Phi \in \mathcal{C}_{\mathcal{N}}(\tau_<)$ , we need to find the corresponding  $\mathcal{N}$ -specific machine  $T^{\text{MOD}[\Gamma]}$ . This seems to be undecidable. Fortunately, we can change form (2) in such a way as to obtain a recursive set of  $\mathcal{L}_{\mathcal{N}}(\tau_<)$ -sentences defining all  $\mathcal{N}$ -complete problems on  $\text{STRUC}[\tau_<]$ .

With each nonempty binary string  $w$ , we associate the following identically false first order sentence  $\psi_w$ :

$$\mathcal{Q}x_1 \dots \mathcal{Q}x_k [x_1 \neq x_1 \wedge \dots \wedge x_k \neq x_k]$$

where  $k = |w|$ ;  $\mathcal{Q}x_i$  denotes either  $\exists x_i$  if  $w[i] = 1$ , or  $\forall x_i$  if  $w[i] = 0$ , for every  $1 \leq i \leq k$ . We say that  $\psi_w$  encodes the binary string  $w$ , and call  $\psi_w$  an encoding sentence. Since such a first order sentence can be considered in a certain sense as a representation of binary strings, we will use this sentence for an appropriate encoding of Turing machines. Let us introduce the following form:

$$(\gamma_{\Gamma, T^{\text{MOD}[\Gamma]}} \wedge \Gamma) \vee (\neg \gamma_{\Gamma, T^{\text{MOD}[\Gamma]}} \wedge \Upsilon_{\tau_<}) \vee \psi_{\langle T^{\text{MOD}[\Gamma]} \rangle} \quad (5)$$

Note that form (5) is logically equivalent to form (2) since  $\psi_{\langle T^{\text{MOD}[\Gamma]} \rangle}$  is identically false. However, in contrast to (2), form (5) allows us to easily construct a logic capturing the complexity class of all  $\mathcal{N}$ -complete problems on all ordered structures. Indeed, let  $\mathcal{C}\mathcal{O}\mathcal{L}_{\mathcal{N}}$  be a mapping from every vocabulary  $\tau_<$  containing  $<$  to  $\mathcal{C}\mathcal{O}\mathcal{L}_{\mathcal{N}}(\tau_<)$ , where  $\mathcal{C}\mathcal{O}\mathcal{L}_{\mathcal{N}}(\tau_<)$  denotes the following set of  $\mathcal{L}_{\mathcal{N}}(\tau_<)$ -sentences of form (5):

$$\{\Phi \in \mathcal{L}_{\mathcal{N}}(\tau_<) \mid \Phi = (\gamma_{\Gamma, T^{\text{MOD}[\Gamma]}} \wedge \Gamma) \vee (\neg \gamma_{\Gamma, T^{\text{MOD}[\Gamma]}} \wedge \Upsilon_{\tau_<}) \vee \psi_{\langle T^{\text{MOD}[\Gamma]} \rangle}; \\ \Gamma \in \mathcal{L}_{\mathcal{N}}(\tau_<), T^{\text{MOD}[\Gamma]} \text{ is a } \mathcal{N}\text{-specific Turing machine}\}.$$

Then, the following theorem holds.

**Theorem 9** *Let  $\mathcal{L}_{\mathcal{N}}$  be a logic capturing a complexity class  $\mathcal{N}$  (among the classes NL, P, coNP, NP, and PSPACE) on  $\text{STRUC}[\tau_<]$  for some fixed vocabulary  $\tau_<$  containing  $<$ . Then,  $\mathcal{C}\mathcal{O}\mathcal{L}_{\mathcal{N}}$  is a decidable fragment of  $\mathcal{L}_{\mathcal{N}}$  that captures the complexity class of all  $\mathcal{N}$ -complete problems on  $\text{STRUC}[\tau_<]$ .*

**PROOF.** Let us show how to effectively decide whether  $\Phi \in \mathcal{C}\mathcal{O}\mathcal{L}_{\mathcal{N}}(\tau_<)$  or not, given an arbitrary  $\Phi \in \mathcal{L}_{\mathcal{N}}(\tau_<)$ . At first, we verify whether  $\Phi$  is of the form  $(\gamma \wedge \Gamma) \vee (\neg \gamma \wedge \Upsilon) \vee \psi$  or not, where  $\Gamma$  and  $\Upsilon$  are both  $\mathcal{L}_{\mathcal{N}}(\tau_<)$ -sentences;  $\gamma$  and  $\psi$  are both first order sentences. If this is not the case, then  $\Phi \notin \mathcal{C}\mathcal{O}\mathcal{L}_{\mathcal{N}}$ . Otherwise, we apply the operator  $T_{\tau_<}$  to  $\Upsilon(R(x))$ , and obtain  $\Upsilon_{\tau_<}$ . If  $\Upsilon \neq \Upsilon_{\tau_<}$ , then  $\Phi \notin \mathcal{C}\mathcal{O}\mathcal{L}_{\mathcal{N}}$ . Otherwise, we check whether  $\psi$  is an encoding sentence  $\psi_w$  or not. If this is not the case, then  $\Phi \notin \mathcal{C}\mathcal{O}\mathcal{L}_{\mathcal{N}}$ . Otherwise, we verify whether  $w$  is a code of some  $\mathcal{N}$ -specific Turing machine  $T^{\text{MOD}[\Gamma]}$  or not. If  $w$  does not encode any  $\mathcal{N}$ -specific Turing machine  $T^{\text{MOD}[\Gamma]}$ , then  $\Phi \notin \mathcal{C}\mathcal{O}\mathcal{L}_{\mathcal{N}}$ . Otherwise, we recover  $T^{\text{MOD}[\Gamma]}$  from its code  $w$ . Then, we construct the characteristic sentence  $\gamma_{\Gamma, T^{\text{MOD}[\Gamma]}}$  for the pair  $(\Gamma, T^{\text{MOD}[\Gamma]})$ . If  $\gamma = \gamma_{\Gamma, T^{\text{MOD}[\Gamma]}}$ , then  $\Phi \in \mathcal{C}\mathcal{O}\mathcal{L}_{\mathcal{N}}$ . Otherwise,  $\Phi \notin \mathcal{C}\mathcal{O}\mathcal{L}_{\mathcal{N}}$ .

Thus, the set  $\mathcal{C}\mathcal{O}\mathcal{L}_{\mathcal{N}}(\tau_{<})$  of  $\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentences is recursive, and  $\mathcal{C}\mathcal{O}\mathcal{L}_{\mathcal{N}}$  is a logic that represents a decidable fragment of  $\mathcal{L}_{\mathcal{N}}$ . Since form (5) is logically equivalent to form (2), Theorem 5 holds for form (5) as well as for form (2). Then, a problem  $\Pi \subseteq \text{STRUC}[\tau_{<}]$  is  $\mathcal{N}$ -complete if and only if there exists a  $\mathcal{C}\mathcal{O}\mathcal{L}_{\mathcal{N}}(\tau_{<})$ -sentence  $\Lambda$  defining  $\Pi$ . This concludes the proof of the theorem.  $\square$

**Corollary 10** *If a logic  $\mathcal{L}_{\mathcal{N}}$  captures a complexity class  $\mathcal{N}$  on all ordered structures, then the logic  $\mathcal{C}\mathcal{O}\mathcal{L}_{\mathcal{N}}$  captures the complexity class of all  $\mathcal{N}$ -complete problems on all ordered structures.*

PROOF. It is immediate from the theorem.  $\square$

Now, we proceed to unordered non-Aristotelian structures. Let us introduce the following form:

$$(\Theta_{\Gamma, T^{\text{MOD}}[\Gamma]} \wedge \Gamma) \vee (\overline{\Theta}_{\Gamma, T^{\text{MOD}}[\Gamma]} \wedge \Upsilon_{\tau}) \vee \psi_{\langle T^{\text{MOD}}[\Gamma] \rangle} \quad (6)$$

Note that form (6) is logically equivalent to form (4) since  $\psi_{\langle T^{\text{MOD}}[\Gamma] \rangle}$  is identically false. However, in contrast to (4), form (6) allows us to easily construct a logic capturing the complexity class of all  $\mathcal{N}$ -complete problems on all non-Aristotelian structures. Indeed, let  $\mathcal{C}\mathcal{L}_{\mathcal{N}}$  be a mapping from every non-Aristotelian vocabulary  $\tau$  to  $\mathcal{C}\mathcal{L}_{\mathcal{N}}(\tau)$ , where  $\mathcal{C}\mathcal{L}_{\mathcal{N}}(\tau)$  denotes the following set of  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentences of form (6):

$$\{\Phi \in \mathcal{L}_{\mathcal{N}}(\tau) \mid \Phi = (\Theta_{\Gamma, T^{\text{MOD}}[\Gamma]} \wedge \Gamma) \vee (\overline{\Theta}_{\Gamma, T^{\text{MOD}}[\Gamma]} \wedge \Upsilon_{\tau}) \vee \psi_{\langle T^{\text{MOD}}[\Gamma] \rangle}; \\ \Gamma \in \mathcal{L}_{\mathcal{N}}(\tau), T^{\text{MOD}}[\Gamma] \text{ is a } \mathcal{N}\text{-specific Turing machine}\}.$$

Then, the following theorem holds.

**Theorem 11** *Let  $\mathcal{L}_{\mathcal{N}}$  be a logic capturing a complexity class  $\mathcal{N}$  (among the classes  $\text{coNP}$ ,  $\text{NP}$ , and  $\text{PSPACE}$ ) on  $\text{STRUC}[\tau]$  for a fixed arbitrary non-Aristotelian vocabulary  $\tau$ . Then,  $\mathcal{C}\mathcal{L}_{\mathcal{N}}$  is a decidable fragment of  $\mathcal{L}_{\mathcal{N}}$  that captures the complexity class of all  $\mathcal{N}$ -complete problems on  $\text{STRUC}[\tau]$ .*

PROOF. Let us show how to effectively decide whether  $\Phi \in \mathcal{C}\mathcal{L}_{\mathcal{N}}(\tau)$  or not, given an arbitrary  $\Phi \in \mathcal{L}_{\mathcal{N}}(\tau)$ . At first, we verify whether  $\Phi$  is of the form  $(\Theta \wedge \Gamma) \vee (\Xi \wedge \Upsilon) \vee \psi$  or not, where  $\Theta$ ,  $\Gamma$ ,  $\Xi$ , and  $\Upsilon$  are  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentences;  $\psi$  is a first order sentence. If this is not the case, then  $\Phi \notin \mathcal{C}\mathcal{L}_{\mathcal{N}}$ . Otherwise, we apply the operator  $T_{\tau}$  to  $\Upsilon(R(x, y))$ , and obtain  $\Upsilon_{\tau}$ . If  $\Upsilon \neq \Upsilon_{\tau}$ , then  $\Phi \notin \mathcal{C}\mathcal{L}_{\mathcal{N}}$ . Otherwise, we check whether  $\psi$  is an encoding sentence  $\psi_w$  or not. If this is not the case, then  $\Phi \notin \mathcal{C}\mathcal{L}_{\mathcal{N}}$ . Otherwise, we verify whether  $w$  is a code of some  $\mathcal{N}$ -specific Turing machine  $T^{\text{MOD}}[\Gamma]$  or not. If  $w$  does not encode any  $\mathcal{N}$ -specific Turing machine  $T^{\text{MOD}}[\Gamma]$ , then  $\Phi \notin \mathcal{C}\mathcal{L}_{\mathcal{N}}$ . Otherwise, we recover  $T^{\text{MOD}}[\Gamma]$  from its code  $w$ . Then, we construct the characteristic sentence  $\Theta_{\Gamma, T^{\text{MOD}}[\Gamma]}$  for the pair  $(\Gamma, T^{\text{MOD}}[\Gamma])$ . Also, we construct the complementary sentence  $\overline{\Theta}_{\Gamma, T^{\text{MOD}}[\Gamma]}$  for  $\Theta_{\Gamma, T^{\text{MOD}}[\Gamma]}$ , given  $T^{\text{MOD}}[\Gamma]$  and  $\Gamma$ . If  $\Theta = \Theta_{\Gamma, T^{\text{MOD}}[\Gamma]}$  and  $\Xi = \overline{\Theta}_{\Gamma, T^{\text{MOD}}[\Gamma]}$ , then  $\Phi \in \mathcal{C}\mathcal{L}_{\mathcal{N}}$ . Otherwise,  $\Phi \notin \mathcal{C}\mathcal{L}_{\mathcal{N}}$ .

Thus, the set  $\mathcal{CL}_{\mathcal{N}}(\tau)$  of  $\mathcal{L}_{\mathcal{N}}(\tau)$ -sentences is recursive, and  $\mathcal{CL}_{\mathcal{N}}$  is a logic that represents a decidable fragment of  $\mathcal{L}_{\mathcal{N}}$ . Since form (6) is logically equivalent to form (4), Theorem 8 holds for form (6) as well as for form (4). Then, a problem  $\Pi \subseteq \text{STRUC}[\tau]$  is  $\mathcal{N}$ -complete if and only if there exists a  $\mathcal{CL}_{\mathcal{N}}(\tau)$ -sentence  $\Lambda$  defining  $\Pi$ . This concludes the proof of the theorem.  $\square$

**Corollary 12** *If a logic  $\mathcal{L}_{\mathcal{N}}$  captures a complexity class  $\mathcal{N}$  on all structures, then the logic  $\mathcal{CL}_{\mathcal{N}}$  captures the complexity class of all  $\mathcal{N}$ -complete problems on all non-Aristotelian structures.*

PROOF. It is immediate from the theorem.  $\square$

## 6 Canonical form and logic for $\text{NP} \cap \text{coNP}$

Let a vocabulary  $\tau_{<}$  containing  $<$  be given. In a very similar way (see section 5), we will show that one cannot recursively enumerate all  $\text{SO}\exists(\tau_{<})$ -sentences defining problems in  $\text{NP} \cap \text{coNP}$ , provided that  $\text{NP} \cap \text{coNP} \neq \text{NP}$ . Let  $\Gamma$  denote a distinguished  $\text{SO}\exists(\tau_{<})$ -sentence that defines some problem in  $\text{NP} \cap \text{coNP}$ . Here the notation  $\Upsilon_{\tau_{<}}$  stands for a  $\text{SO}\exists(\tau_{<})$ -sentence defining a NP-complete problem. Let us consider a  $\text{SO}\exists(\tau_{<})$ -sentence  $\Phi$  of the form  $(\gamma_G \wedge \Gamma) \vee (\neg\gamma_G \wedge \Upsilon_{\tau_{<}})$ . Note that if the statement  $L(G) = \Sigma^*$  holds, then  $\gamma_G$  is logically valid. It follows that  $\Phi$  is logically equivalent to  $\Gamma$ , and  $\Phi$  defines a problem in  $\text{NP} \cap \text{coNP}$  in this case. If the statement  $L(G) \neq \Sigma^*$  holds, then  $\text{MOD}[\gamma_G]$  is finite. Therefore, the set  $\text{MOD}[\Upsilon_{\tau_{<}}]$  differs from the set  $\text{MOD}[\Phi]$  only in a finite set of  $\tau_{<}$ -structures, i.e.  $\text{MOD}[\Upsilon_{\tau_{<}}] \setminus \text{MOD}[\Phi]$  and  $\text{MOD}[\Phi] \setminus \text{MOD}[\Upsilon_{\tau_{<}}]$  are both finite. Then,  $\Phi$  defines a NP-complete problem that cannot be in  $\text{NP} \cap \text{coNP}$ , on the assumption  $\text{NP} \cap \text{coNP} \neq \text{NP}$ . Thus,  $\Phi$  defines a problem in  $\text{NP} \cap \text{coNP}$  if and only if the statement  $L(G) = \Sigma^*$  holds.

Let an arbitrary context-free grammar  $G$  with the terminal alphabet  $\Sigma$  be given. We construct the  $\text{SO}\exists(\tau_{<})$ -sentence  $(\gamma_G \wedge \Gamma) \vee (\neg\gamma_G \wedge \Upsilon_{\tau_{<}})$  for this grammar  $G$ . Suppose that we recursively enumerate all  $\text{SO}\exists(\tau_{<})$ -sentences  $\Phi_1, \Phi_2, \dots$  that define problems in  $\text{NP} \cap \text{coNP}$ . If we find  $\Phi_i$  such that  $\Phi_i = (\gamma_G \wedge \Gamma) \vee (\neg\gamma_G \wedge \Upsilon_{\tau_{<}})$ , then we have  $L(G) = \Sigma^*$ . However, this contradicts the fact that the statement  $L(G) = \Sigma^*$  is algorithmically undecidable (see section 5). Thus, one cannot recursively enumerate all  $\text{SO}\exists(\tau_{<})$ -sentences that define problems on  $\text{STRUC}[\tau_{<}]$  in  $\text{NP} \cap \text{coNP}$ , provided that  $\text{NP} \cap \text{coNP} \neq \text{NP}$ .

Then, we state the following question. What is it about a  $\text{SO}\exists$ -sentence that makes its defined problem be in  $\text{NP} \cap \text{coNP}$ ? We will answer this question in the affirmative.

Gurevich [10] conjectured that no logic captures the complexity class  $\text{NP} \cap \text{coNP}$ . The conjecture is based on the fact that the existence of such a logic (in the sense of Gurevich) for  $\text{NP} \cap \text{coNP}$  implies the existence of a complete problem in  $\text{NP} \cap \text{coNP}$ . However, there does not exist such a complete problem relative to a certain oracle [18]. Therefore,  $\text{NP} \cap \text{coNP}$  very likely has no complete problem.

Nevertheless, it is not necessary for a logic to imply the existence of a complete problem. For example, no complete problem is known in the complexity class PH, whereas second order logic captures PH (see [19]).

We can extend our approach beyond complete problems. A canonical form (similar to form (6)) will be developed to define problems in  $\text{NP} \cap \text{coNP}$ . Moreover, we will develop a logic that captures  $\text{NP} \cap \text{coNP}$  and does not require the existence of any complete problem in  $\text{NP} \cap \text{coNP}$ .

Let  $\tau$  be a fixed vocabulary, where  $\tau$  may be any (including Aristotelian) vocabulary. Then, with each pair  $(\Lambda, \Gamma)$  of  $\text{SO}\exists(\tau)$ -sentences, we associate the following set of  $\tau$ -structures:

$$S_{\Lambda, \Gamma} \triangleq \{ A \in \text{STRUC}[\tau] \mid (\forall B \in \text{STRUC}[\tau]) [ \|B\| > \ell^{(2)}(|\langle A \rangle|) \vee (B \models \Lambda \Leftrightarrow B \not\models \Gamma) ] \}.$$

Note that the set  $S_{\Lambda, \Gamma}$  has very similar properties as the previously considered set  $S_{\Gamma, \text{TMOD}[\Gamma]}$ . We have the following theorem.

**Theorem 13**  $S_{\Lambda, \Gamma} \in \text{DTIME}[n]$ .

**PROOF.** It is similar to the proof of Theorem 6 in many respects. Let us decide whether  $A \in S_{\Lambda, \Gamma}$  or not, given a  $\tau$ -structure  $A$ . Let  $\hat{n}$  denote  $|\langle A \rangle|$ .

First, we need to compute the number  $\ell^{(2)}(\hat{n})$ . It is obvious that we can find  $\ell^{(2)}(\hat{n})$  in time  $O(\hat{n})$ .

Second, we enumerate all  $\tau$ -structures  $B \in \text{STRUC}[\tau]$  such that  $\|B\| \leq \ell^{(2)}(\hat{n})$ . Since  $|\langle B \rangle| \leq p(\|B\|)$ , the value  $|\langle B \rangle|$  is bounded above by  $p(\log \log \hat{n})$ , where  $p$  is a polynomial dependent on the vocabulary  $\tau$ . Therefore, this enumeration takes time  $O(2^{p(\log \log \hat{n})})$ . Then, for each  $\tau$ -structure  $B$  of the enumeration, we need to verify the following condition:

$$B \models \Lambda \Leftrightarrow B \not\models \Gamma. \tag{7}$$

If condition (7) is not satisfied for some  $\tau$ -structure  $B$  of the enumeration, then the input  $\tau$ -structure  $A$  does not belong to  $S_{\Lambda, \Gamma}$ . Otherwise,  $A \in S_{\Lambda, \Gamma}$ . Now, let us estimate the running time for verifying condition (7).

We can verify (7) for one  $\tau$ -structure  $B$  in time  $O(2^{p'(\log \log \hat{n})})$ , where  $p'$  is a polynomial. This follows from the fact that any model-checking problem in  $\text{NP} \cap \text{coNP}$  is decidable at most in exponential time.

Consequently, the total time to verify (7) for all  $\tau$ -structures  $B$  of the enumeration is  $O(2^{p(\log \log \hat{n})} \cdot 2^{p'(\log \log \hat{n})})$ , or, in short,  $O(2^{p''(\log \log \hat{n})})$ , where

$p'' = p + p'$ . Since  $\lim_{\hat{n} \rightarrow \infty} \frac{2^{p''(\log \log \hat{n})}}{\hat{n}} = 0$  for any polynomial  $p''$ , this time can be approximately estimated at most as  $O(\hat{n})$ .

Thus, the overall running time consists of the time for computing  $\ell^{(2)}(\hat{n})$  and of the time for verifying (7) on all  $\tau$ -structures  $B$  of the enumeration. The both of these times are estimated as  $O(\hat{n})$ . Therefore, so is the overall running time. This concludes the proof of the theorem.  $\square$

**Corollary 14** *Given a pair  $(\Lambda, \Gamma)$  of  $\text{SO}\exists(\tau)$ -sentences, one can construct a  $\text{SO}\exists(\tau)$ -sentence that defines  $S_{\Lambda, \Gamma}$ .*

PROOF. It follows from the statement  $S_{\Lambda, \Gamma} \in \text{DTIME}[n]$ .  $\square$

Then, by  $\Theta_{\Lambda, \Gamma}$  we denote the constructed  $\text{SO}\exists(\tau)$ -sentence defining the model class  $S_{\Lambda, \Gamma}$ . This  $\Theta_{\Lambda, \Gamma}$  characterizes  $\Lambda$  and  $\Gamma$  in the following way: if  $\Theta_{\Lambda, \Gamma}$  is logically valid, then  $\text{MOD}[\Lambda]$  and  $\text{MOD}[\Gamma]$  are both in  $\text{NP} \cap \text{coNP}$ . We call the  $\text{SO}\exists(\tau)$ -sentence  $\Theta_{\Lambda, \Gamma}$  the characteristic sentence for the pair  $(\Lambda, \Gamma)$ . Then, the following theorem holds.

**Theorem 15** *Let  $\Pi \subseteq \text{STRUC}[\tau]$  be a model class. Then,  $\Pi$  is in  $\text{NP} \cap \text{coNP}$  if and only if there exists a  $\text{SO}\exists(\tau)$ -sentence  $\Omega$  such that  $\text{MOD}[\Omega] = \Pi$  and  $\Omega$  is of the form*

$$(\Theta_{\Lambda, \Gamma} \wedge \Gamma) \vee \psi_{\langle \Lambda \rangle} \quad (8)$$

where  $\Lambda$  and  $\Gamma$  are both  $\text{SO}\exists(\tau)$ -sentences;  $\Theta_{\Lambda, \Gamma}$  the characteristic sentence for  $(\Lambda, \Gamma)$ ;  $\psi_{\langle \Lambda \rangle}$  the sentence encoding the binary string  $\langle \Lambda \rangle$ .

PROOF. For short, we denote form (8) by  $\Phi_{\Lambda, \Gamma}$ .

First, we will prove that  $\Phi_{\Lambda, \Gamma}$  defines a problem in  $\text{NP} \cap \text{coNP}$  for any pair  $(\Lambda, \Gamma)$  of  $\text{SO}\exists(\tau)$ -sentences. Suppose that  $\text{MOD}[\Gamma] = \text{STRUC}[\tau] \setminus \text{MOD}[\Lambda]$ . Then,  $\text{MOD}[\Gamma] \in \text{coNP}$ , and, therefore, we have  $\text{MOD}[\Gamma] \in \text{NP} \cap \text{coNP}$ . Note that  $\Theta_{\Lambda, \Gamma}$  is logically valid in this case. Then,  $\Phi_{\Lambda, \Gamma}$  is logically equivalent to  $\Gamma$ . It follows that  $\Phi_{\Lambda, \Gamma}$  defines the problem  $\text{MOD}[\Gamma]$  in  $\text{NP} \cap \text{coNP}$ . Now, suppose that  $\text{MOD}[\Gamma] \neq \text{STRUC}[\tau] \setminus \text{MOD}[\Lambda]$ . Then,  $\text{MOD}[\Theta_{\Lambda, \Gamma}]$  is finite, and so is  $\text{MOD}[\Phi_{\Lambda, \Gamma}]$ . Since the set  $\text{MOD}[\Phi_{\Lambda, \Gamma}]$  is finite,  $\text{MOD}[\Phi_{\Lambda, \Gamma}] \in \text{NP} \cap \text{coNP}$  in this case as well.

Second, we will prove that any problem in  $\text{NP} \cap \text{coNP}$  can be defined by means of the form  $\Phi_{\Lambda, \Gamma}$ . Let  $\Gamma$  be a  $\text{SO}\exists(\tau)$ -sentence defining an arbitrary problem in  $\text{NP} \cap \text{coNP}$ . Then,  $\text{MOD}[\Gamma]$  and  $\text{STRUC}[\tau] \setminus \text{MOD}[\Gamma]$  are both in  $\text{NP}$ . Therefore, there exists a  $\text{SO}\exists(\tau)$ -sentence  $\Lambda$  that defines the model class  $\text{STRUC}[\tau] \setminus \text{MOD}[\Gamma]$ . Note that the characteristic sentence  $\Theta_{\Lambda, \Gamma}$  for the pair  $(\Lambda, \Gamma)$  is logically valid in this case. Moreover,  $\Gamma$  is logically equivalent to  $\Phi_{\Lambda, \Gamma}$ . Since  $\Gamma$  defines a problem in  $\text{NP} \cap \text{coNP}$ , so does  $\Phi_{\Lambda, \Gamma}$ . This concludes the proof of the theorem.  $\square$

Thus, form (8) can serve as a canonical form for  $\text{NP} \cap \text{coNP}$ . Now, we can use form (8) in order to develop a logic capturing  $\text{NP} \cap \text{coNP}$ . By  $\mathcal{L}_{\text{NP} \cap \text{coNP}}$  we mean a mapping from every vocabulary  $\tau$  to  $\mathcal{L}_{\text{NP} \cap \text{coNP}}(\tau)$ , where  $\mathcal{L}_{\text{NP} \cap \text{coNP}}(\tau)$  denotes the following set of  $\text{SO}\exists(\tau)$ -sentences:

$$\{\Phi \in \text{SO}\exists(\tau) \mid \Phi = (\Theta_{\Lambda, \Gamma} \wedge \Gamma) \vee \psi_{\langle \Lambda \rangle}; \Gamma \in \text{SO}\exists(\tau), \Lambda \in \text{SO}\exists(\tau)\}.$$

Then, the following theorem holds.

**Theorem 16**  *$\mathcal{L}_{\text{NP} \cap \text{coNP}}$  is a decidable fragment of  $\text{SO}\exists$  logic that captures  $\text{NP} \cap \text{coNP}$ .*

PROOF. It is similar to the proof of Theorem 11. Let us fix an arbitrary vocabulary  $\tau$ . We will show how to effectively decide whether  $\Phi \in \mathcal{L}_{\text{NP} \cap \text{coNP}}(\tau)$  or not, given an arbitrary  $\Phi \in \text{SO}\exists(\tau)$ . At first, we verify whether  $\Phi$  is of the form  $(\Theta \wedge \Gamma) \vee \psi$  or not, where  $\Gamma$  and  $\Theta$  are both  $\text{SO}\exists(\tau)$ -sentences;  $\psi$  is a first order sentence. If this is not the case, then  $\Phi \notin \mathcal{L}_{\text{NP} \cap \text{coNP}}$ . Otherwise, we check whether  $\psi$  is an encoding sentence  $\psi_w$  or not. If this is not the case, then  $\Phi \notin \mathcal{L}_{\text{NP} \cap \text{coNP}}$ . Otherwise, we check whether  $w$  encodes some well-formed  $\text{SO}\exists(\tau)$ -sentence  $\Lambda$  or not. If this is not the case, then  $\Phi \notin \mathcal{L}_{\text{NP} \cap \text{coNP}}$ . Otherwise, we recover  $\Lambda$  from its code  $w$ . Then, we construct the characteristic sentence  $\Theta_{\Lambda, \Gamma}$  for the pair  $(\Lambda, \Gamma)$ . If  $\Theta = \Theta_{\Lambda, \Gamma}$ , then  $\Phi \in \mathcal{L}_{\text{NP} \cap \text{coNP}}$ . Otherwise,  $\Phi \notin \mathcal{L}_{\text{NP} \cap \text{coNP}}$ .

Thus, the set  $\mathcal{L}_{\text{NP} \cap \text{coNP}}(\tau)$  of  $\text{SO}\exists(\tau)$ -sentences is recursive, and  $\mathcal{L}_{\text{NP} \cap \text{coNP}}$  is a logic that represents a decidable fragment of  $\text{SO}\exists$ . Note that Theorem 15 implies that a problem  $\Pi \subseteq \text{STRUC}[\tau]$  is in  $\text{NP} \cap \text{coNP}$  if and only if there exists a  $\mathcal{L}_{\text{NP} \cap \text{coNP}}(\tau)$ -sentence  $\Omega$  defining  $\Pi$ . This concludes the proof of the theorem.  $\square$

## 7 Concluding remarks

We have developed canonical forms for problems that are complete via Turing reductions. Also, we have shown that any complete problem can be easily defined by one of these forms. Besides, we have provided an evidence that there cannot be any complete problem on Aristotelian structures in the complexity classes P, coNP, NP, and PSPACE.

Logics for complete problems in the complexity classes NL, P, coNP, NP, and PSPACE have been developed on the basis of the canonical form (5) that defines these problems on ordered structures. On the other hand, logics for complete problems in the complexity classes coNP, NP, and PSPACE have been developed on the basis of the canonical form (6) that defines these problems on unordered non-Aristotelian structures. It is very likely that analogous canonical forms can be also developed to construct logics for complete problems in other complexity classes.

Besides, we have extended our approach beyond complete problems. Using a similar form, we have developed a logic that captures the complexity class  $\text{NP} \cap \text{coNP}$  which very likely contains no complete problem. Note that a recursive enumeration of all problems in  $\text{NP} \cap \text{coNP}$  was considered to be difficult (see, for instance, [5, 16]). Up to our knowledge, no such recursive enumeration was known till now. Dawar [5] pointed out: "...the natural set of witnesses for  $\text{NP} \cap \text{coNP}$  is not recursively enumerable. Thus, finding a recursively enumerable set of witnesses would require a fundamentally new characterization of the class and would be a major breakthrough in complexity theory.". Nevertheless, using the logic  $\mathcal{L}_{\text{NP} \cap \text{coNP}}$  to capture  $\text{NP} \cap \text{coNP}$ , we recursively enumerate all problems in  $\text{NP} \cap \text{coNP}$  by enumerating the sentences of this logic.

In conclusion, we have modified a fragment of Immerman's diagram [11] in respect to the complexity classes from P to PSPACE, as shown in Figure 2 (cf.

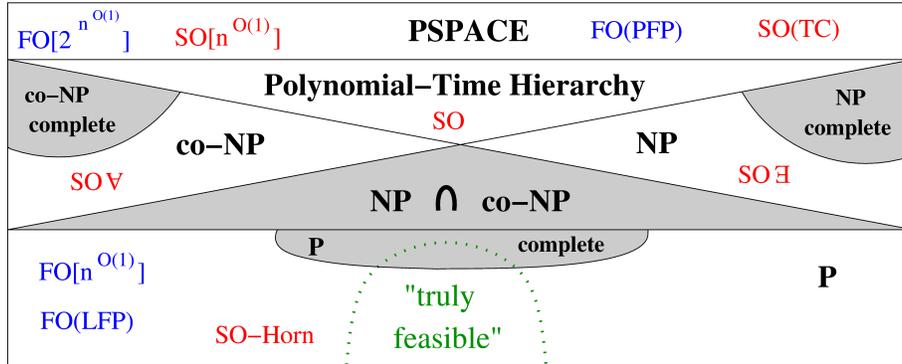


Figure 2: The World of Descriptive and Computational Complexity from P to PSPACE: shaded areas indicate our developed logics for NP-complete problems, coNP-complete problems, P-complete problems, and  $NP \cap coNP$ .

Figure 1). For purposes of clarity, in the diagram we have permitted ourself to shade areas depicting the following complexity classes: NP-complete problems, coNP-complete problems, P-complete problems, and  $NP \cap coNP$  for which we have developed logics for the first time. Moreover, for the complexity classes of PSPACE-complete problems and NL-complete problems which are not depicted in Immerman’s diagram, we have developed logics as well.

## References

- [1] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.
- [2] P. Berman. Relationship between density and deterministic complexity of NP-complete languages. In *Proceedings of the Fifth Colloquium on Automata, Languages and Programming*, pp. 63–71, 1978.
- [3] B. Bonet and N. Borges. Syntactic characterizations of completeness using duals and operators. *Logic Journal of the IGPL*, **20**, 266–282, 2012.
- [4] J. Cai and D. Sivakumar. Sparse hard sets for P: resolution of a conjecture of Hartmanis. *Journal of Computer and System Sciences*, **58**, 280–296, 1999.
- [5] A. Dawar. On complete problems, relativizations and logics for complexity classes. In *Fields of Logic and Computation*, Vol. 6300 of *Lecture Notes in Computer Science*, A. Blass, N. Dershowitz, and W. Reisig, eds, pp. 201–207. Springer, 2010.

- [6] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In *Complexity of Computation*, R. Karp, ed., pp. 27–41. SIAM-AMS Proceedings 7, 1974.
- [7] S. Ginsburg. *The Mathematical Theory of Context-Free Languages*. McGraw-Hill Book Company, New York, 1966.
- [8] E. Grädel, P.G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M.Y. Vardi, Y. Venema, and S. Weinstein. *Finite Model Theory and Its Applications*. Springer, 2007.
- [9] P. Grünwald and P. Vitányi. Algorithmic information theory. In *Handbook of the Philosophy of Science. Volume 8: Philosophy of Information*, P. Adriaans and J. van Benthem, eds, pp. 281–317. Elsevier B.V., 2008.
- [10] Y. Gurevich. Logic and the challenge of computer science. In *Current Trends in Theoretical Computer Science*, E. Börger, ed., pp. 1–57. Computer Science Press, Rockville, 1988.
- [11] N. Immerman. *Descriptive Complexity*. Springer, 1998.
- [12] R. E. Ladner and N. A. Lynch. Relativization of questions about log space computability. *Mathematical Systems Theory*, **10**, 19–32, 1976.
- [13] J. Lukasiewicz. *Aristotle’s Syllogistic from the Standpoint of Modern Formal Logic*. Oxford University Press, 1957.
- [14] J. A. Medina and N. Immerman. A syntactic characterization of NP-completeness. In *Proceedings of the 9th IEEE Symposium on Logic in Computer Science*, pp. 241–250, 1994.
- [15] V. G. Naidenko. To the problem  $P \stackrel{?}{=} NP$ . *Doklady Mathematics*, **53**, 411–412, 1996.
- [16] C. R. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, **48**, 498–532, 1994.
- [17] D. Richerby. Logical characterizations of PSPACE. In *Computer Science Logic: 18th International Workshop, CSL 2004*, Vol. 3210 of *Lecture Notes in Computer Science*, J. Marcinkowski and A. Tarlecki, eds, pp. 370–384. Springer, 2004.
- [18] M. Sipser. On relativization and the existence of complete sets. In *Proceedings of the 9th International Colloquium on Automata, Languages and Programming*, Vol. 140 of *Lecture Notes in Computer Science*, M. Nielsen and E.M. Schmidt, eds, pp. 523–531. Springer, 1982.
- [19] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, **3**, 1–22, 1977.