The Expressive Power of Henkin Quantifiers with Dualization

Siła wyrazu kwantyfikatorów Henkina z dualizacją

Leszek Aleksander Kołodziejczyk

June 9, 2002

Master's Thesis
Instytut Filozofii, Uniwersytet Warszawski

Supervisor
dr hab. Marcin Mostowski
Instytut Filozofii, Uniwersytet Warszawski

$$\begin{matrix} \forall & \exists \\ \forall & \exists \end{matrix}$$

Abstract

The thesis discusses an extension of the logic with Henkin quantifiers (a survey of Henkin quantifers may be found in [Krynicki–Mostowski 1995a]). The extension, known as logic with branched quantifiers with dualization (LBD), differs from the usual logic with Henkin quantifiers in that the set of available quantifiers is additionally closed on a dualization operator (where the dualization of a quantifier Q is defined as $\neg Q\neg$).

In [Mostowski 1987a] it was shown that LBD is semantically equivalent to second order logic. In this thesis we use a naturally defined notion of dualization depth of a quantifier to examine the semantical force of sublogics of LBD. We prove that for any natural number n, the class of simple positive formulae with quantifiers of dualization depth at most n is semantically equivalent to the class Σ^1_{n+1}. We also show that the expressive power of the full logic with quantifiers of dualization depth at most n is bounded by the class Δ^1_{n+2}.

One of the corollaries of our results is that Henkin quantifiers of dualization depth at most n are a natural class of quantifiers capturing the complexity class $\mathbf{\Sigma^P_{n+1}}$ of the so–called polynomial hierarchy.

Streszczenie

Praca omawia pewne rozszerzenie logiki z kwantyfikatorami Henkina (przegląd stanu wiedzy na temat kwantyfikatorów Henkina znaleźć można w pracy [Krynicki–Mostowski 1995a]). Rozszerzenie to, znane jako logika z kwantyfikatorami Henkina z dualizacją (LBD), różni się od zwykłej logiki z kwantyfikatorami Henkina tym, że zbiór dostępnych kwantyfikatorów jest dodatkowo domknięty na operator dualizacji (gdzie dualizacją kwantyfikatora Q jest $\neg Q\neg$).

M. Mostowski wykazał w [Mostowski 1987a], że LBD jest semantycznie równoważna logice drugiego rzędu. W niniejszej pracy używamy naturalnie zdefiniowanego pojęcia głębokości dualizacji kwantyfikatora do badania semantycznej mocy podlogik LBD. Dowodzimy, że dla każdej liczby naturalnej n, klasa prostych formuł pozytywnych z kwantyfikatorami z dualizacją głębokości co najwyżej n jest semantycznie rownoważna klasie Σ^1_{n+1}. Pokazujemy także, że siła wyrazu pełnej logiki z kwantyfikatorami z dualizacją głębokości co najwyżej n jest ograniczona z góry przez klasę Δ^1_{n+2}.

Jednym z wniosków z naszych wyników jest to, że kwantyfikatory Henkina z dualizacją głębokości co najwyżej n są naturalną klasą kwantyfikatorów chwytającą klasę złożoności $\mathbf{\Sigma^P_{n+1}}$ z tak zwanej hierarchii wielomianowej.

Acknowledgments

We would like to thank: our thesis supervisor, Marcin Mostowski, for suggesting to us the topic of the thesis, and for providing us with constant guidance and help without which we certainly would not have obtained our results; Konrad Zdanowski, who is the co–author of one of the two main theorems of the thesis; and our Parents, for everything.

Contents

1 Introduction

In [Henkin 1961], Leon Henkin introduced a certain extension of first order quantification. The linear quantifiers of first order logic do not suffice to express all possible dependencies between the quantified variables. Henkin's idea was to overcome this limitation by allowing explicit mention of dependencies between variables. The logic with his quantifiers — originally referred to as partially ordered quantifiers, nowadays called *Henkin quantifiers* or *branched quantifiers*— turned out to be an interesting, and intensively studied, enrichment of first order logic[1].

The reasons for studying Henkin quantifiers are numerous and varied. Apart from their purely logical properties, some of which will be discussed in Chapter 2 of this thesis, Henkin quantifiers have a number of features which seem to be important from a philosophical point of view. It is, for example, sometimes held (see [Hintikka 1974]) that they are indispensable for the proper reconstruction of the logical forms of some sentences of natural language. Furthermore, they considerably enhance the expressive power of our language (they capture a large fragment of second order logic) at apparently no ontological cost (they bind only first order variables).

This last feature of Henkin quantifiers becomes almost spectacular once we introduce an additional, very natural operation known as dualization. The so–called *Henkin quantifiers with dualization* at first sight appear to be a minor variant of the usual Henkin quantifiers. This impression is misleading, however: in [Mostowski 1987a] it was proved that the logic with Henkin quantifiers with dualization (LBD[2]) is semantically equivalent to full second order logic.

LBD is the central topic of this paper. In Chapter 3, we study its equivalence with second order logic in some detail. We prove two new results on the structure of this equivalence. In particular, we show that in LBD there is an exact counterpart of the Σ^1_n hierarchy of second order logic. We also obtain an upper bound for the expressive power of certain fragments of LBD[3].

In recent years, another motivation for the study of generalized quantifiers has come from the direction of computational complexity theory. Beginning with Fagin's theorem (see [Fagin 1974]), we have learned that many

[1] A comprehensive survey of the research on the logic with Henkin quantifiers may be found in the paper [Krynicki–Mostowski 1995a]. The paper [Mostowski–Zdanowski 2002] is a survey on the logic with Henkin quantifiers in finite models. Additionally, [Mostowski 1995] discusses proof systems for Henkin quantifiers and a non–standard semantics known as the weak semantics.

[2] **L**ogic of **B**ranched quantifiers with **D**ualization.

[3] The upper bound is a joint result with Konrad Zdanowski.

questions of complexity theory are equivalent to questions concerning the expressibility of properties of finite models in various extensions of first order logic. The discovery of this phenomenon has led to the development of the so–called descriptive complexity theory, which attempts to tackle the problems of complexity theory by means of logical considerations. Henkin quantifiers have played a key role in these considerations since the publication of the paper [Blass–Gurevich 1986]. Among the results contained there was the observation that the classes of finite models definable by simple positive formulae with Henkin quantifiers are exactly those which are **NP**.

In Chapter 4, the final chapter of the thesis, we apply the complexity–theoretical approach to LBD. Our results from Chapter 3 allow us to determine the complexity classes corresponding to Henkin quantifiers with dualization. This, in turn, makes it possible to gain additional insight into the behaviour of LBD in finite models.

The paper closes with an Appendix on the ideas and facts in complexity theory essential for understanding our results.

2 Basic definitions and facts

This chapter contains some basic definitions and facts on Henkin quantifiers. As we concentrate almost exclusively on notions and theorems necessary for understanding our results, our exposition is by no means complete. A (much) more thorough treatment of Henkin quantifiers may be found in [Krynicki–Mostowski 1995a].

2.1 Logic with Henkin quantifiers

Definition 2.1 *A Henkin prefix is a triple $Q = (A_Q, E_Q, D_Q)$, where A_Q and E_Q are disjoint finite sets of variables (the universal and existential variables of Q, respectively) and $D_Q \subseteq A_Q \times E_Q$. D_Q is the dependency relation of Q. Whenever $(x, y) \in D_Q$, we say that the existential variable y depends on the universal variable x in Q. We say that Q is linear if there is a linear ordering $<$ on $A_Q \cup E_Q$ such that $(x, y) \in D_Q$ iff $x < y$, for $x \in A_Q, y \in E_Q$.*

Henkin prefixes are usually written down not as triples, but in a more elegant and readable form. We write them as rows of standard first order prefixes, in such a way that each existential variable depends exactly on those universal variables which are on the left of it and in the same row. For example, the prefix $\mathsf{H} = (\{x, z\}, \{y, w\}, \{(x, y), (z, w)\})$ is:

$$\begin{matrix} \forall x & \exists y \\ \forall z & \exists w. \end{matrix}$$

In the case of more complex prefixes, the notational convention is slightly modified. We may use parentheses to divide prefixes into blocks. It is then the rule that an existential variable depends on all universal variables which are on the left of its block. Within a single block, the dependencies are determined as previously. For example, the prefix

$$(\{x_1, x_2, z_1, z_2\}, \{y_1, y_2, w_1, w_2\},$$
$$\{(x_1, y_1), (x_2, y_2), (x_1, w_1), (x_1, w_2), (x_2, w_1), (x_2, w_2), (z_1, w_1), (z_2, w_2)\})$$

would be written down as:

$$\begin{pmatrix} \forall x_1 & \exists y_1 \\ \forall x_2 & \exists y_2 \end{pmatrix} \begin{pmatrix} \forall z_1 & \exists w_1 \\ \forall z_2 & \exists w_2 \end{pmatrix}.$$

From now on, we shall not distinguish precisely between prefixes and quantifiers. Generally, "quantifier" is both a syntactical and a semantical concept, whereas "prefix" is a purely syntactical one. A prefix may be thought of as a quantifier once we disregard its specific variables.

$\mathcal{H}$ is the set of all Henkin prefixes. We follow the standard convention and use $\mathcal{H}$ also to denote the class of all Henkin quantifiers. L^*_σ is the logic obtained from first order logic in vocabulary σ by adding all Henkin quantifiers.

We now proceed to define the set of formulae and the satisfaction relation for L^*_σ. The set of terms of L^*_σ is the same as for first order logic. The set of formulae of L^*_σ is defined as follows:

Definition 2.2 *$\mathcal{F}_{L^*_\sigma}$ is the smallest set satisfying the following conditions:*

- *for any terms t_1, t_2 of vocabulary σ, $\ulcorner t_1 = t_2 \urcorner \in \mathcal{F}_{L^*_\sigma}$,*
- *for any R, k–ary relation in σ, and for any terms $t_1, \ldots, t_k$ of vocabulary σ, $\ulcorner R(t_1, \ldots, t_k) \urcorner \in \mathcal{F}_{L^*_\sigma}$*
- *for any ψ and φ, if $\psi, \varphi \in \mathcal{F}_{L^*_\sigma}$, then $\ulcorner \neg\psi \urcorner \in \mathcal{F}_{L^*_\sigma}$ and $\ulcorner (\psi \wedge \varphi) \urcorner \in \mathcal{F}_{L^*_\sigma}$,*
- *for any variable x and φ, if $\varphi \in \mathcal{F}_{L^*_\sigma}$, then $\ulcorner \exists x \varphi \urcorner \in \mathcal{F}_{L^*_\sigma}$,*
- *for any $Q \in \mathcal{H}$ and φ, if $\varphi \in \mathcal{F}_{L^*_\sigma}$, then $\ulcorner Q\varphi \urcorner \in \mathcal{F}_{L^*_\sigma}$.*

To define the semantics for Henkin quantifiers, we need the concept of skolemization of a formula relative to a prefix:

Definition 2.3 *Let $Q \in \mathcal{H}$. Let $x_1, \ldots, x_n$ be the universal and $y_1, \ldots, y_k$ the existential variables of Q. For $i = 1, \ldots, k$, let $\mathbf{x}_i$ be the sequence of all the universal variables on which y_i depends in Q. Then the skolemization $sk(Q, \varphi)$ of $Q\varphi$ relative to Q is the result of substituting $f_i(\mathbf{x}_i)$ for y_i in φ (for $i = 1, \ldots, k$). The function symbols $f_1, \ldots, f_k$ have to be new and mutually distinct. They are called the Skolem functions introduced by skolemization of $Q\varphi$ relative to Q.*

Definition 2.4 *The satisfaction relation $\models_{L^*_\sigma}$ is defined inductively as for first logic, with the following inductive condition added:*

$$M \models_{L^*_{\sigma'}} Q\varphi\,[\mathbf{a}] \text{ if and only if there are operations } F_1, \ldots, F_k \text{ on } |M| \text{ such that } (M, F_1, \ldots, F_k) \models_{L^*_{\sigma'}} \forall \mathbf{x} sk(Q, \varphi)\,[\mathbf{a}],$$

where $F_1, \ldots, F_k$ interpret the respective Skolem functions introduced by skolemization of $Q\varphi$ relative to Q, σ' is the extension of σ by these Skolem functions, and $\mathbf{x}$ is the sequence of all the universal variables of Q.

It is natural to ask whether the logic with Henkin quantifiers is semantically stronger than first order logic. The answer is positive, as is evidenced by the following example.

Example. The Ehrenfeucht sentence[4] is a sentence of $L^*_\emptyset$ of the form

$$\exists t \begin{matrix} \forall x & \exists y \\ \forall z & \exists w \end{matrix} ((x = z \equiv y = w) \wedge (t \neq y)).$$

The Ehrenfeucht sentence is true exactly in infinite models.

Indeed, by the definition of the semantics for $L^*_\emptyset$, the Ehrenfeucht sentence is true in a model M if and only if there is a one-to-one function on $|M|$ which is not onto, i.e. exactly if M is infinite (in Dedekind's sense). □

The semantics for Henkin quantifiers defined above is known as the functional semantics. This semantics has a certain drawback: its correctness depends on the correctness of the skolemization technique, which in turn relies on the Axiom of Choice. For this reason, in [Mostowski 1987] an alternative semantics, called the relational semantics, was proposed.

To develop the relational semantics, we will have to adopt a somewhat different definition of the set of Henkin prefixes.

Definition 2.5 (The set of quantifier prefixes $PREF$) *The set $PREF$ and the natural mapping $PREF \longrightarrow \mathcal{H}$ are defined by parallel induction as follows:*

- *for any variable x: $\forall x \in PREF$, $A_{\forall x} = \{x\}$, $E_{\forall x} = \emptyset$, and $D_{\forall x} = \emptyset$; also $\exists x \in PREF$, $A_{\exists x} = \emptyset$, $E_{\exists x} = \{x\}$, and $D_{\exists x} = \emptyset$;*
- *for any $Q, Q' \in PREF$ such that the sets $A_Q, A_{Q'}, E_Q$, and $E_{Q'}$ are mutually disjoint: $\begin{matrix} Q \\ Q' \end{matrix} \in PREF$, $A_{\begin{matrix} Q \\ Q' \end{matrix}} = A_Q \cup A_{Q'}$, $E_{\begin{matrix} Q \\ Q' \end{matrix}} = E_Q \cup E_{Q'}$, and $D_{\begin{matrix} Q \\ Q' \end{matrix}} = D_Q \cup D_{Q'}$;*
  also $QQ' \in PREF$, $A_{QQ'} = A_Q \cup A_{Q'}$, $E_{QQ'} = E_Q \cup E_{Q'}$, and $D_{QQ'} = D_Q \cup D_{Q'} \cup (A_Q \times E_{Q'})$.

For prefixes Q, Q', the prefix $\begin{matrix} Q \\ Q' \end{matrix}$ is called the vertical, while QQ' — the horizontal composition of Q and Q'. It should be noted that the natural mapping from $PREF$ to $\mathcal{H}$, defined by $Q \mapsto (A_Q, E_Q, D_Q)$, is *not* surjective.

[4]This sentence was discovered by Andrzej Ehrenfeucht and first appeared in [Henkin 1961].

For a vocabulary σ, the set of formulae L^r_σ (where L^r is the logic with quantifiers from $PREF$) is defined analogously to L^*_σ, with $\mathcal{H}$ replaced by $PREF$. The satisfaction relation for the relational semantics is then defined thus:

Definition 2.6 *The satisfaction relation $\models_{L^r_\sigma}$ is defined inductively as for first order logic, with the following inductive conditions added:*

$$M \models_{L^r_\sigma} (QQ')\varphi\,[\mathbf{a}] \text{ if and only if } M \models_{L^r_\sigma} Q(Q'\varphi)\,[\mathbf{a}];$$

$$M \models_{L^r_\sigma} \begin{matrix} Q \\ Q' \end{matrix} \varphi\,[\mathbf{a}] \text{ if and only if there are relations } R, R' \text{ on } |M| \text{ such that}$$
$$(M, R, R') \models_{L^r_{\sigma'}} QP(\mathbf{x}) \wedge Q'P'(\mathbf{x}') \wedge \forall \mathbf{x}\mathbf{x}'(P(\bar{x}) \wedge P'(\bar{x}') \Rightarrow \varphi)\,[\mathbf{a}],$$

where R, R' interpret the predicates P, P' (respectively), P and P' do not occur in σ, the vocabulary σ' is the extension of σ by these predicates, and $\mathbf{x}, \mathbf{x}'$ are sequences of all the variables of Q and Q', respectively.

The question arises whether the relational semantics is equivalent to the functional semantics. To see that such an equivalence indeed does hold, we need another definition.

Definition 2.7 *The L^*–form of an L^r–formula φ is a formula φ^* obtained from φ by replacing all occurences of quantifier prefixes from $PREF$ by prefixes from $\mathcal{H}$ assigned to them by means of the natural mapping.*

Assuming the Axiom of Choice, we now have a theorem establishing the equivalence of the two semantics.

Theorem 2.8 *The following holds:*

(i) if φ is an L^r–formula and φ^* is its L^*–form, then for any model M and any valuation $\mathbf{a}$ in M we have: $M \models \varphi\,[\mathbf{a}]$ if and only if $M \models \varphi^*\,[\mathbf{a}]$.

(ii) every L^*–formula is equivalent to the L^*–form of some L^r–formula.

2.2 Comparing logics

We have already seen that the logic with Henkin quantifiers is more powerful than ordinary first order logic. As a matter of fact, we can characterize its expressive power more precisely. This is carried out by comparing the logic with Henkin quantifiers to appropriate fragments of second order logic.

We recall the most important concepts used in comparing logics. Any logic is a functor (proper class function) L which assigns to every vocabulary

σ a pair $(F_{\sigma L}, \models_{\sigma L})$, where $F_{\sigma L}$ is the set of L–sentences of vocabulary σ and $\models_{\sigma L}$ is a (proper class) relation between models of vocabulary σ and elements of $F_{\sigma L}$ (the truth relation). It is assumed that logics satisfy some general conditions. In this paper, we consider only natural logics which obviously satisfy these conditions[5].

Definition 2.9 *For a sentence $\varphi \in F_{\sigma L}$ we define $Mod_{\sigma L}(\varphi)$ as the class of models of vocabulary σ in which φ is true (that is, in which $M \models_{\sigma L} \varphi$ holds). Two sentences φ, ψ are logically equivalent if $Mod_{\sigma L}(\varphi) = Mod_{\sigma L}(\psi)$ for any vocabulary σ containing the vocabularies of φ and ψ.*

We will skip the indices L and σ whenever they will be clear from the context.

Definition 2.10 *Let L, L' be logics. We define:*

- *$L \leq L'$ (L is semantically weaker or equivalent to L') if for every vocabulary σ and every $\varphi \in F_{\sigma L}$ there exists an L'–sentence $\psi \in F_{\sigma L'}$ logically equivalent to φ.*
- *$L \equiv L'$ (L is semantically equivalent to L') if $L \leq L'$ and $L' \leq L$.*
- *$L < L'$ (L is semantically weaker than L') if $L \leq L'$ but not $L' \leq L$.*

Within second order logic, there is a well–known classification of formulae. A second order formula in normal form (i.e. with all second order quantifiers at the beginning, followed by a series of first order quantifiers and a quantifier–free formula[6]) is Σ^1_1 if all of its second order quantifiers are existential, Π^1_1 if all of its second order quantifiers are universal. A formula is Σ^1_{n+1} if it consists of a series of second order existential quantifiers followed by a Π^1_n–formula; it is Π^1_{n+1} if it consists of a series of second order universal quantifiers followed by a Σ^1_n–formula. This classification can also be applied to classes of models.

Definition 2.11 *Let K be a class of models of vocabulary σ (closed on isomorphisms). K is:*

- *Σ^1_n if there is a Σ^1_n–sentence φ in vocabulary σ such that $K = Mod_\sigma(\varphi)$.*

[5]For the general concept of (*model–theoretic* or *abstract*) logic see e.g. the book [Chang–Keisler 1990].

[6]Assuming AC, every second order formula is equivalent to a formula in such a normal form.

- Π^1_n *if there is a* Π^1_n*–sentence* φ *in vocabulary* σ *such that* $K = Mod_\sigma(\varphi)$.
- Δ^1_n *if it is both* Σ^1_n *and* Π^1_n.

It is fairly easy to notice that for every $n \in \omega - \{0\}$, Π^1_n contains exactly those classes of models which correspond to negations of Σ^1_n–sentences (and vice versa). By adding superfluous quantifiers, we also see that for all $n < m$, $\Sigma^1_n \subseteq \Sigma^1_m$ and $\Pi^1_n \subseteq \Pi^1_m$. Furthermore, $\Sigma^1_n \subseteq \Sigma^1_{n+1}$, $\Pi^1_n \subseteq \Pi^1_{n+1}$, and $\Sigma^1_n \subseteq \Delta^1_{n+1} \subseteq \Sigma^1_{n+1}$ (the same holds for Δ– and Π–classes). In fact, all of these inclusions are, in general, strict. Thus, we obtain a hierarchy of classes of models (which can also be viewed as a hierarchy of logics), sometimes referred to as the second order hierarchy.

The semantical strength of L^r is comparable to the lower levels of the second order hierarchy.

Definition 2.12 *For a set* $\mathcal{Q}$ *of quantifiers, the set* $N^+(\mathcal{Q})$ *of simple formulae with quantifiers from* $\mathcal{Q}$ *consists of all formulae of the form* $Q\psi$*, where* $Q \in \mathcal{Q}$ *and* ψ *is first order.*

Fact 2.13 $N^+(\mathcal{H}) \equiv N^+(PREF)$.

The folowing theorems give a precise description of the semantical power of the logic with Henkin quantifiers.

Theorem 2.14 ([Enderton 1970], [Walkoe 1970]) *There is an effective procedure assigning to a* Σ^1_1*–formula* φ *a formula* $\tilde{\varphi} \in N^+(\mathcal{H})$ *such that* $\tilde{\varphi}$ *is semantically equivalent to* φ.

Theorem 2.15 ([Enderton 1970]) *For any* L^**–formula* φ *we can effectively find* Π^1_2*– and* Σ^1_2*–formulae semantically equivalent to* φ.

Theorem 2.16 ([Mostowski 1991a]) *There is a* Δ^1_2*–class of models not expressible in* L^*.

Summarizing these results, we obtain:

Corollary 2.17 *The following holds*[7]*:*

(i) $N^+(PREF) \equiv \Sigma^1_1$;

(ii) $L^r < \Delta^1_2$.

[7]The notation $L < \Delta^1_n$ has not been formally defined, but it should be understood as (proper) inclusion between appropriate families of classes of models.

Hence, the logic with Henkin quantifiers captures a significant though limited fragment of second order logic. It turns out, however, that there exists an apparently slight modification of L^r which enhances its expressive power even further. This modification will be the topic of the remaining two chapters.

3 Henkin quantifiers with dualization

We are now ready to discuss the main topic of this paper, a certain modification of Henkin quantifiers. The modification consists in allowing dualization of quantifier prefixes.

3.1 Dualizing Henkin quantifiers

We think of a quantifier Q as *dual* to a quantifier Q' if Q is equivalent to $\neg Q' \neg$. The most basic example is $\exists$, which is dual to $\forall$ (actually, since the duality relation is obviously symmetrical, also $\forall$ is dual to $\exists$). It is interesting that the only Henkin quantifiers which have dual Henkin quantifiers are the linear ones, i.e. those already definable in first order logic (see e.g. [Mostowski 1991]). One might therefore wonder about the effects of extending the set of quantifier prefixes by introducing an explicit dualization operator.

Definition 3.1 *(The set of prefixes with dualization $PREF^d$). The set $PREF^d$ is defined analogously to $PREF$, with the following inductive condition added:*

- *for $Q \in PREF^d$: $Q^d \in PREF^d$.*

From now on, we write "DHQ" to denote "Henkin quantifier with dualization". The logic with DHQs is called LBD. For any vocabulary σ, the set of LBD–formulae of vocabulary σ is defined in the natural way. The satisfaction relation is also defined as expected:

Definition 3.2 *The satisfaction relation for LBD is defined as for L^r, with the following inductive condition added:*

$$M \models_{LBD_\sigma} (Q^d)\varphi\,[\mathbf{a}] \text{ if and only if } M \models_{LBD_\sigma} \neg Q \neg \varphi\,[\mathbf{a}].$$

There is an important normal–form theorem for LBD, which *does not hold* for the logic with ordinary Henkin quantifiers:

Theorem 3.3 ([Mostowski 1987a], [Mostowski 1987b]) *For every LBD–formula φ there is $Q \in PREF^d$ and a quantifier free LBD–formula ψ of the same vocabulary such that $\models (\varphi \iff Q\psi)$, i.e. φ and $Q\psi$ are logically equivalent.*

We have already hinted at the fact that LBD is more expressive than L^r. It is not very difficult to find examples for this.

Example. It is known — see [Mostowski 1991] — that a sentence of L^* (and hence L^r) in empty vocabulary has an infinite model if and only if it has a countable model. Therefore, there is no sentence of L^r in empty vocabulary true exactly in countable models. However, there are such sentences in LBD. An example is the sentence ξ in $N^+(PREF^d)$ consisting of the prefix

$$\exists z \begin{pmatrix} \forall x_1 & \exists y_1 \\ \forall x_2 & \exists y_2 \end{pmatrix} \begin{pmatrix} \forall t_1 & \exists u_1 \\ \forall t_2 & \exists u_2 \\ \exists w & \end{pmatrix}^d$$

followed by a quantifier–free formula

$$(x_1 = x_2 \Rightarrow y_1 = y_2) \wedge ((t_1 = t_2 \Rightarrow u_1 = u_2) \Rightarrow \vartheta),$$

where ϑ is:

$$\big\{x_1 \neq y_1 \wedge (y_1 = y_2 \Rightarrow x_1 = x_2) \wedge\ y_1 \neq z\big\} \wedge$$

$$\wedge\big\{[(t_1 = z \Rightarrow t_1 = u_1) \wedge (t_1 = x_1 \wedge t_2 = y_1 \wedge u_1 = x_2 \Rightarrow u_2 = y_2)] \Rightarrow$$

$$\Rightarrow (t_1 = w \wedge w = u_1)\big\}.$$

The sentence ξ is a considerably simplified version of the sentence obtained by applying the translation procedure used in the proof of theorem 3.10 (see below) to a second order sentence expressing countability.

We will outline the proof that ξ is indeed true exactly in countable models. We use P, R, S, T as relational variables and s, f, possibly with indices, as unary functional variables.

By definition of the semantics for LBD, ξ is logically equivalent to ξ_1:

$$\exists P \exists R \exists z \big\{ \begin{matrix} \forall x_1 & \exists y_1 \\ \forall x_2 & \exists y_2 \end{matrix} (P(x_1, x_2, y_1, y_2)) \wedge \neg \begin{pmatrix} \forall t_1 & \exists u_1 \\ \forall t_2 & \exists u_2 \\ \exists w & \end{pmatrix} (\neg R(t_1, t_2, u_1, u_2, w)) \wedge$$

$$\wedge\, \forall x_1 \forall x_2 \forall y_1 \forall y_2 \forall t_1 \forall t_2 \forall u_1 \forall u_2 \forall w [P(x_1, x_2, y_1, y_2) \wedge R(t_1, t_2, u_1, u_2, w) \Rightarrow$$

$$\Rightarrow ((x_1 = x_2 \Rightarrow y_1 = y_2) \wedge ((t_1 = t_2 \Rightarrow u_1 = u_2) \Rightarrow \vartheta))]\big\}.$$

We need to translate ξ_1 into a logically equivalent second order formula true exactly in countable models. We shall limit ourselves to the basic steps

of the translation procedure and omit the details. The procedure is based on applying a series of tricks. One of these consists in replacing formulae such as

$$\exists S\Big\{\begin{matrix}\forall x\ \exists y\\ \forall z\ \exists w\end{matrix}\ S(x,z,y,w) \wedge \forall x\forall y\forall z\forall w[S(x,z,y,w) \Rightarrow$$

$$\Rightarrow ((x = z \Rightarrow y = w) \wedge \varphi(x,z,y,w))]\Big\}$$

by the equivalent $\exists f\forall x\forall z\varphi(x,z,f(x),f(z))$. Another trick makes use of the equivalence of $\forall x\varphi(x)$ with $\forall f\forall x\varphi(f(x))$.

Let us now observe that in ξ_1 we may take $P(x_1,x_2,y_1,y_2)$ to be simply $y_1 = s(x_1) \wedge y_2 = s(x_2)$ for an appropriate function s. Thus, when we eliminate the non–linear Henkin quantifiers, we see that ξ_1 is logically equivalent to ξ_2:

$$\exists s\exists R\exists z\Big\{\neg\exists f_1\exists f_2\exists w\forall t_1\forall t_2(\neg R(t_1,t_2,w,f_1(t_1),f_2(t_2))) \wedge$$

$$\wedge \forall f_1\forall f_2\forall x_1\forall x_2\forall t_1\forall t_2\forall w$$

$$\Big[R(t_1,t_2,w,f_1(t_1),f_2(t_2)) \wedge (t_1 = t_2 \Rightarrow f_1(t_1) = f_2(t_2)) \Rightarrow$$

$$\Rightarrow \{x_1 \neq s(x_1) \wedge (s(x_1) = s(x_2) \Rightarrow x_1 = x_2) \wedge s(x_1) \neq z \wedge$$

$$\wedge [f_1(z) = z \wedge (t_2 = s(t_1) \Rightarrow f_2(t_2) = s(f_1(t_1))) \Rightarrow$$

$$\Rightarrow (t_1 = w = f_1(t_1))]\}\Big]\Big\}.$$

It is fairly easy to see that ξ_2 is logically equivalent to ξ_3:

$$\exists s\exists T\exists z\Big\{\neg\exists f\exists w\forall t_1(\neg T(t_1,w,f(t_1))) \wedge$$

$$\wedge \forall f\forall x_1\forall x_2\forall t_1\forall w\Big[T(t_1,w,f(t_1)) \Rightarrow$$

$$\{x_1 \neq s(x_1) \wedge (s(x_1) = s(x_2) \Rightarrow x_1 = x_2) \wedge s(x_1) \neq z \wedge$$

$$\wedge [f(z) = z \wedge (f(s(t_1)) = s(f(t_1))) \Rightarrow (t_1 = w = f(t_1))]\}\Big]\Big\}.$$

Finally, ξ_3 is logically equivalent to:

$$\exists s\exists z\{\text{“}s \textit{ satisfies the successor axioms (is injective, has no fixed point,}$$

$$\textit{and never has the zero element as value) with zero } z\text{”} \wedge$$

$$\forall f[f(z) = z \wedge \forall x(f(s(x)) = s(f(x))) \Rightarrow \forall t(f(t) = t)]\}.$$

This last sentence is obviously true exactly in models of cardinality ω. □

Actually, the growth in expressive power is enormous, as is shown by the next theorem:

Theorem 3.4 ([Mostowski 1987a], [Mostowski 1987b]) [8] *$LBD \equiv SO$ (second order logic).*

Thus, LBD is a curious logic which, on the one hand, uses only quantification over first order variables, on the other hand, is semantically equivalent to full second order logic![9]

In the remainder of this chapter and in the next chapter — i.e. in the core of this paper — we delve deeper into the structure of this equivalence, proving some more detailed results and pointing out a few interesting consequences.

Before that, however, we prove two lemmas on DHQs as generalized quantifiers (we will need them later on). First, we recall the notion of a generalized quantifier[10]:

Definition 3.5 *A generalized quantifier Q of type $(n_1, \ldots, n_k)$ is a functor which assigns to every set X a k–ary relation $Q(X)$ between relations on X such that if $(R_1, \ldots, R_k) \in Q(X)$, then R_i is an n_i–ary relation on X, for $i = 1, \ldots, k$. Q has to be preserved by bijections, i.e. if $f : X \longrightarrow Y$ is a bijection then for every k–tuple $(R_1, \ldots, R_k)$ of relations on X:*

$$(R_1, \ldots, R_k) \in Q(X) \text{ if and only if } (fR_1, \ldots, fR_k) \in Q(Y),$$

where $fR = \{(f(x_1), \ldots, f(x_j)) : (x_1, \ldots, x_j) \in R\}$, for $R \subseteq X^j$.

A quantifier Q of type $(n_1, \ldots, n_k)$ binds $n_1 + \ldots + n_k$ first order variables and k formulae. For formulae $\psi_1, \ldots, \psi_k$, a model M, and a valuation $\mathbf{a}$ in $|M|$, we have

$$M \models Q\mathbf{x}(\psi_1, \ldots, \psi_k)\ [\mathbf{a}] \text{ if and only if } (\psi_1^{M,\mathbf{a},\mathbf{x}_1}, \ldots, \psi_k^{M,\mathbf{a},\mathbf{x}_k}) \in Q(|M|),$$

where $\varphi^{M,\mathbf{a},\mathbf{y}}$ is the relation defined by the formula φ in M under valuation $\mathbf{a}$ with respect to the variables $\mathbf{y}$, and the $\mathbf{x}_i$'s are sequences of distinct variables, each of length n_i (for $i = 1, \ldots, k$), from which the sequence $\mathbf{x}$ arises by concatenation.

Both ordinary Henkin quantifiers and DHQs always bind only one formula and thus are of type (m), for some $m \in \omega$. We say that a DHQ of type (m) is m–ary. The interpretations of Henkin quantifiers and DHQs as generalized quantifiers are, for the most part, obvious. We note only that if Q is an m–ary quantifier, X is a set and $R \subseteq X^m$, then $R \in Q^d(X)$ if and only if $X^m - R \notin Q(X)$.

We may now state and prove the lemmas.

---

[8] A proof of this theorem may also be found in [Mostowski 1991].

[9] A non–standard semantics for branched quantifiers, under which the logic with branched quantifiers itself is equivalent to SO, was introduced in the paper [Harel 1979].

[10] For a short overview of generalized quantifiers, and a bibliography, we refer to [Krynicki–Mostowski 1995].

Lemma 3.6 *DHQs are monotone, i.e. if Q is an m–ary DHQ, X is any set, and S, T are m–ary relations on X such that $S \subseteq T$, then: if $S \in Q(X)$, then $T \in Q(X)$.*

Proof. We fix a set X and use induction on quantifier structure.

The thesis is obvious for $\forall$ and $\exists$. Assume Q is monotone and take $S \subseteq T \subseteq X^m$, where $S \in Q^d(X)$. In that case $X^m - S \notin Q(X)$, so, by the inductive assumption, $X^m - T \notin Q(X)$, which gives $T \in Q^d(X)$. Hence, Q^d is also monotone.

Now assume that Q and Q' are monotone, where Q is m_1–ary, Q' is m_2–ary, and let $m = m_1 + m_2$.

Consider $\begin{matrix} Q \\ Q' \end{matrix}$. Take $S \subseteq T \subseteq X^m$, $S \in \begin{matrix} Q \\ Q' \end{matrix}(X)$. Any relations R_1, R_2 which witness that $S \in \begin{matrix} Q \\ Q' \end{matrix}(X)$ also witness that $T \in \begin{matrix} Q \\ Q' \end{matrix}(X)$.

Now consider QQ'. Take $S \subseteq T \subseteq X^m$, $S \in QQ'(X)$. For an m–ary relation R on X, denote by A_R the set $\{\mathbf{x} \in X^{m_1} : \{\mathbf{y} \in X^{m_2} : (\mathbf{x}, \mathbf{y}) \in R\} \in Q'(X)\}$[11]. It is easy to observe that $S \in QQ'(X)$ is equivalent to $A_S \in Q(X)$. But Q' is monotone, which means $A_S \subseteq A_T$. Again, Q is monotone, so $A_T \in Q(X)$ and therefore $T \in QQ'(X)$. $\square$

Lemma 3.7 *If Q is an m–ary DHQ, then for every non–empty set X, $\emptyset \notin Q(X)$ and $X^m \in Q(X)$.*

Proof. We fix X and use induction on quantifier structure.

The thesis is obvious for $\forall$ and $\exists$. Assume the thesis holds for Q. Since $X^m - \emptyset = X^m \in Q(X)$, then $\emptyset \notin Q^d(X)$. Again, since $X^m - X^m = \emptyset \notin Q(X)$, then $X^m \in Q^d(X)$. Hence, the thesis holds for Q^d.

Now assume the thesis holds for Q and Q', which are m_1– and m_2–ary, respectively, and let $m = m_1 + m_2$.

Consider $\begin{matrix} Q \\ Q' \end{matrix}$. Assume that there are R and R' — subsets of X^{m_1} and X^{m_2}, respectively — which witness that $\emptyset \in \begin{matrix} Q \\ Q' \end{matrix}(X)$, in other words, such that $R \in Q(X)$, $R' \in Q'(X)$, and $R \times R' \subseteq \emptyset$. Then either R or R' is empty. Assume without loss of generality that $R = \emptyset$. In that case, by the inductive assumption, $R \notin Q(X)$, which contradicts the choice of R. So

[11] We use freely the natural identification of $X^{m_1} \times X^{m_2}$ with $X^{m_1+m_2}$.

$\emptyset \notin \begin{matrix} Q \\ Q' \end{matrix}(X)$. On the other hand, any R, R' such that $R \in Q(X), R' \in Q'(X)$ satisfy $R \times R' \subseteq X^m$ and thus witness that $X^m \in \begin{matrix} Q \\ Q' \end{matrix}(X)$.

Now consider QQ'. As $A_\emptyset$ (we use the notation from the previous proof) is — by the inductive assumption — empty and does not belong to $Q(X)$, we have $\emptyset \notin QQ'(X)$. Likewise, A_{X^m} is all of X^{m_1}, which belongs to $Q(X)$, so $X^m \in QQ'(X)$. $\square$

Lemma 3.7 will be necessary in the proof of theorem 3.11. We use there the fact that if Q is a DHQ and ψ is any formula such that the variables of Q and ψ are disjoint, then $Q\varphi \wedge \psi$ is logically equivalent to $Q(\varphi \wedge \psi)$ and that the same holds for disjunctions. For this to be true, it is sufficient that Q satisfies the thesis of the lemma[12].

3.2 Simple formulae with limited dualization depth

Once the correspondence between LBD and second order logic is established, it is only natural to ask about its details. In particular, one might attempt to generalize Enderton and Walkoe's theorems (theorems 2.14 and 2.15) and ask whether in LBD there is any counterpart to the second order hierarchy. Finding such a counterpart would not only be interesting for its own sake, but would also be of some relevance e.g. for descriptive complexity theory.

The obvious way to search for such a counterpart is to restrict one's attention to fragments of LBD in which only quantifiers of limited dualization depth are allowed.

Definition 3.8 *The dualization depth (d) of DHQs is defined by induction on quantifier structure:*

- $d(Q) = 0$ *if* $Q \in PREF$, *i.e. if Q is a Henkin quantifier without dualization*;
- $d(Q^d) = d(Q) + 1$;
- $d(QQ') = d\left(\begin{matrix} Q \\ Q' \end{matrix}\right) = max(\{d(Q), d(Q')\})$.

[12]To see that the logical equivalence of $Q\varphi \wedge \psi$ and $Q(\varphi \wedge \psi)$ does not have to hold in general, take for example a tautology φ, an inconsistent sentence ψ, and Q such that for some set X, $\emptyset \in Q(X)$. In that case, $Q\varphi \wedge \psi$ is always false, whereas $Q(\varphi \wedge \psi)$ is true in some models.

Definition 3.9 *For $n \in \omega$, the set $PREF^d_n$ is a subset of $PREF^d$ consisting of those quantifiers Q for which $d(Q) \leq n$.*

The logic formed by restricting LBD to formulae with all quantifiers in $PREF^d_n$ is denoted by LBD_n.

In [Mostowski 1991] it is stated as a corollary of the proof of theorem 3.4 that for every $n \in \omega$, $\Sigma^1_n \leq N^+(PREF^d_{n-1})$. It will now be our aim to prove the converse, and thus to obtain a suitable generalization of theorem 2.14, or, more precisely, of the first part of corollary 2.17.

We prove the following:

Theorem 3.10 *For any $n \in \omega$, we have $N^+(PREF^d_n) \equiv \Sigma^1_{n+1}$.*

Proof. The proof is by induction on n.

Since $PREF^d_0 = PREF$, the case $n = 0$ is simply the first part of corollary 2.17. The $(\leq)$ direction is obvious due to fact 2.13, the $(\geq)$ direction follows from theorem 2.14.

Now assume that $N^+(PREF^d_n) \equiv \Sigma^1_{n+1}$. We need to prove that $N^+(PREF^d_{n+1}) \equiv \Sigma^1_{n+2}$.

$(\leq)$ What must be shown is that every formula of the form $Q\varphi$, where $Q \in PREF^d_{n+1}$ and φ is first order, is equivalent to some Σ^1_{n+2}–formula. We prove — by induction on quantifier structure — that for every $Q \in PREF^d_{n+1}$ we have:

(*) if φ is a first order formula, then $Q\varphi$ is equivalent to a Σ^1_{n+2}–formula.

The thesis (*) holds for quantifiers in $PREF^d_n$. To see this, notice that by the normal form theorem for first order logic, a first order formula φ is equivalent to a formula of the form $\tilde{Q}\tilde{\varphi}$, where $\tilde{Q}$ is a linear quantifier (and thus belongs to $PREF^d_0$), while $\tilde{\varphi}$ is quantifier–free. Consequently, $Q\varphi$, for $Q \in PREF^d_n$, is equivalent to $Q\tilde{Q}\tilde{\varphi}$, which is a formula in $N^+(PREF^d_n)$ and thus — by the inductive assumption — is equivalent even to some Σ^1_{n+1}–formula.

To complete the inductive proof, we have to show that (*) holds also for dualizations of quantifiers in $PREF^d_n$ and that it is preserved by vertical and horizontal composition of quantifiers. This suffices because the set $PREF^d_{n+1}$ is the closure of the set of all quantifiers in $PREF^d_n$ and their dualizations on vertical and horizontal composition.

Let us first show that (*) holds for dualizations of quantifiers in $PREF^d_n$. Take $Q^d\varphi$, where $Q \in PREF^d_n$ and φ is first order. This is equivalent to

$\neg Q\neg\varphi$, which in turn is equivalent to some Π^1_{n+1}–formula (since $\neg\varphi$ is first order), and thus also to some Σ^1_{n+2}–formula.

Now let (*) hold for Q, Q' and consider $\begin{matrix} Q \\ Q' \end{matrix}\ \varphi$ with φ first order. By definition, $\begin{matrix} Q \\ Q' \end{matrix}\ \varphi$ is equivalent to $\exists R\exists R'(Q\bar{x}R\bar{x}\wedge Q'\bar{x}'R'\bar{x}'\wedge\forall\bar{x}\bar{x}'(R\bar{x}\wedge R'\bar{x}'\Rightarrow\varphi))$. In the latter formula, the first two of the three conjuncts are (up to equivalence) Σ^1_{n+2} by the inductive assumption, and the third is first order, i.e. also Σ^1_{n+2}. The whole conjunction is therefore in Σ^1_{n+2}, and adding the two existential relational quantifiers at the beginning keeps us within this class.

Finally, assume (*) holds for Q, Q', and consider QQ'. Let φ be a first order formula. Assume also, for the sake of simplicity, that $QQ'\varphi$ is a sentence (the general case is easily reducible to this one). Then the following sentences are logically equivalent:

(i) $QQ'\varphi$,

(ii) $Q(Q'\varphi)$,

(iii) $\exists R[\forall\bar{x}(R\bar{x}\Longleftrightarrow Q'\bar{y}\varphi(\bar{x}\bar{y}))\wedge Q\bar{x}R\bar{x}]$,

(iv) $\exists R[\forall\bar{x}(R\bar{x}\Rightarrow Q'\bar{y}\varphi(\bar{x}\bar{y}))\wedge Q\bar{x}R\bar{x}]$.

The equivalences (i)$\Longleftrightarrow$(ii) and (ii)$\Longleftrightarrow$(iii) are obvious, so we show only (iii)$\Longleftrightarrow$(iv). Of course, (iii)$\Rightarrow$(iv) holds. On the other hand, assume that (iv) is true in some model M and consider a relation R on M such that $\forall\bar{x}(R\bar{x}\Rightarrow Q'\bar{y}\varphi(\bar{x}\bar{y}))$ holds in M. Let S be the relation defined in M by $Q'\bar{y}\varphi(\bar{x}\bar{y})$. By (iv), we have $R\subseteq S$, but also $R\in Q(|M|)$. Hence, by the monotonicity of Q (lemma 3.6), $S\in Q(|M|)$, and therefore S witnesses that (iii) is true in M.

It remains to observe that (iv) is (up to equivalence) Σ^1_{n+2}. This sentence is a conjunction preceded by an existential second order quantifier, so it is enough to check that both of the conjuncts are Σ^1_{n+2}, which easily follows from the inductive assumption.

($\geq$) To show that every Σ^1_{n+2}–formula is equivalent to some formula in $N^+(PREF^d_{n+1})$, we shall apply a generalized version of the procedure used in proving theorem 2.14.

Let ξ be a Σ^1_{n+2}–formula. Since we may assume that all of our second order quantifiers are functional, ξ is of the form $\exists\bar{f}_1\forall\bar{f}_2\ldots\Xi\bar{f}_{n+2}\varphi$, where φ is first order and Ξ is $\forall$ or $\exists$ depending on the parity of n. We first observe that $\forall\bar{f}_2\ldots\Xi\bar{f}_{n+2}\varphi$ is equivalent to $\neg\exists\bar{f}_2\ldots\Xi^d\bar{f}_{n+2}\neg\varphi$, which is — by the inductive assumption — equivalent to $\neg Q_1\psi$ (for some $Q_1\in PREF^d_n$ and quantifier–free ψ), and thus also to $Q^d_1\neg\psi$; and Q^d_1 belongs to $PREF^d_{n+1}$. Therefore, all we need to do now is eliminate the functional quantifiers from

the initial sequence $\exists \bar{f}_1$.

This can be done on a "one by one" basis. We will show that there is a formula in $N^+(PREF^d_{n+1})$ equivalent to $\exists f Q \vartheta$, where $Q \in PREF^d_{n+1}$ and ϑ is quantifier–free. Moreover, the first order part of this formula will also be quantifier–free.

Let $f(t_{11}, \ldots, t_{1m}), \ldots, f(t_{k1}, \ldots, t_{km})$ be all occurences, taken as tokens, of f in ϑ. Without loss of generality, we may assume that the t_{ij}'s are terms not containing any occurences of f[13]. In that case, a suitable $N^+(PREF^d_{n+1})$–formula equivalent to $\exists f Q \vartheta$ is:

$$\begin{array}{l} \forall x_{11} \ldots \forall x_{1m} \exists y_1 \\ \qquad \vdots \\ \forall x_{k1} \ldots \forall x_{km} \exists y_k \\ \qquad Q \end{array} \qquad [\mu \wedge ((t_{11} = x_{11} \wedge \ldots \wedge t_{km} = x_{km}) \Rightarrow \vartheta')],$$

where μ is $((x_{11} = x_{21} \wedge \ldots \wedge x_{1m} = x_{2m} \Rightarrow y_1 = y_2) \wedge \ldots \wedge (x_{11} = x_{k1} \wedge \ldots \wedge x_{1m} = x_{km} \Rightarrow y_1 = y_k)$, whereas ϑ' is obtained from ϑ by replacing: first, the t_{ij}'s with the variables x_{ij}, and then, the terms $f(x_{i1}, \ldots, x_{im})$ by the variables y_i (for $i = 1, \ldots, k$; $j = 1, \ldots, m$). $\square$

The argument in the $(\geq)$ direction of the proof is the one originally used to prove theorem 3.4.

3.3 Logics with limited dualization depth

We have managed to obtain an exact estimate of the semantical force of simple formulae with DHQs of limited dualization depth. This is a generalization of the first part of corollary 2.17. Another interesting question concerning the expressive power of DHQs is whether any corresponding estimate can be found for whole logics with limited dualization depth, i.e. whether the second part of corollary 2.17 also allows a similar generalization.

A partial positive answer is provided by the next theorem.

Theorem 3.11 (with Konrad Zdanowski) *For any $n \in \omega$, we have $LBD_n \leq \Delta^1_{n+2}$.*

Proof. The proof goes by induction on n.

[13]To see that there really is no loss of generality, observe that for example $\eta(f(t))$, where t is some complex term, may be replaced by $\exists x(x = t \wedge \eta(f(x))$. The extra quantifier may then be moved to the beginning of ϑ and absorbed by Q.

The case for $n = 0$ is a weaker version of the second part of corollary 2.17. We assume that the thesis is true for n and show that it is also true for $n + 1$. More exactly, we show that:

(#) every LBD_{n+1}–formula φ is equivalent to some formula of the form $(\neg)Q_1(\neg)Q_2 \ldots (\neg)Q_k\psi$, where $Q_i \in PREF^d_{n+1}$ (for $i = 1, \ldots, k$), ψ is quantifier–free, and the $(\neg)$'s stand for either negations or nothing at all.

Let us first see why proving (#) will be enough. Observe that by replacing negations with the d–operator where appropriate, $(\neg)Q_1(\neg)Q_2 \ldots (\neg)Q_k$ may be turned into a sequence of k quantifiers from $PREF^d_{n+2}$ (plus possibly one negation at the end), which is equivalent to one quantifier from $PREF^d_{n+2}$, namely the horizontal composition of these quantifiers (plus possibly the negation). Thus, (#) would imply $LBD_{n+1} \leq N^+(PREF^d_{n+2})$, which by theorem 3.10 gives $LBD_{n+1} \leq \Sigma_{n+3}$. Since LBD_{n+1} is closed on negation, and Δ^1_{n+3} is the largest subclass of Σ^1_{n+3} which is closed on negation, this would yield $LBD_{n+1} \leq \Delta^1_{n+3}$, which is exactly what we need.

By the inductive assumption, (#) holds for formulae in LBD_n. Indeed, the inductive assumption implies $LBD_n \leq N^+(PREF^d_{n+1})$, and any formula in $N^+(PREF^d_{n+1})$ is of the form required in (#).

Obviously, assuming that φ satisfies (#), also $\neg\varphi$ and $Q\varphi$ (for $Q \in PREF^d_{n+1}$) satisfy (#).

Finally, assume φ and ϑ satisfy (#). We will prove that $\varphi \wedge \vartheta$ also satisfies (#). By (#), φ is equivalent to $(\neg)Q_1(\neg)Q_2 \ldots (\neg)Q_k\psi$ and ϑ is equivalent to $(\neg)Q'_1(\neg)Q'_2 \ldots (\neg)Q'_l\psi'$ for some k, l. We replace negations by dualizations to get that φ is equivalent to $\tilde{Q}_1 \ldots \tilde{Q}_k(\neg)\psi$, where the $\tilde{Q}_i$'s are now from $PREF^d_{n+2}$. Thus, φ is also equivalent to $Q_\varphi(\neg)\psi$, where Q_φ is the horizontal composition of all $\tilde{Q}_i$'s. Similarly, ϑ is equivalent to $Q_\vartheta(\neg)\psi'$.

Hence, $\varphi \wedge \vartheta$ is equivalent to $Q_\varphi(\neg)\psi \wedge Q_\vartheta(\neg)\psi'$. Renaming bound variables if necessary, we may assume that Q_φ does not contain any variables from ψ' and that Q_ϑ does not contain any variables from ψ. In that case, $Q_\varphi(\neg)\psi \wedge Q_\vartheta(\neg)\psi'$ is equivalent to $Q_\varphi Q_\vartheta[(\neg)\psi \wedge (\neg)\psi']$ (for this last equivalence to hold, it is enough that both of the quantifiers satisfy the thesis of lemma 3.7). "Unwinding" the dualizations (that is, replacing the dualizations back by negations where appropriate), we get that $\varphi \wedge \vartheta$ is equivalent to $(\neg)Q_1 \ldots (\neg)Q_k(\neg)Q'_1 \ldots (\neg)Q'_l(\neg)[(\neg)\psi \wedge (\neg)\psi']$. $\square$

The special case of theorem 3.11 for $n = 0$ was originally proved by Herbert Enderton in [Enderton 1970] as theorem 2.15. Enderton's proof essentially uses a fairly sophisticated second order tautology. Once we have

the notion of dualization, a simpler proof can be given, as our inductive argument from the proof of theorem 3.11 can be modified so as to prove the case $n = 0$ directly.

We remark that theorem 3.11 generalizes only a weaker version of the result we have for L^r, since we have not proved $LBD_n < \Delta^1_{n+2}$. As a matter of fact, Marcin Mostowski has observed that by using the method of truth definitions one can show that $LBD_n < \Delta^1_{n+2}$ indeed does hold (in infinite models). We do not prove this result here; see the abstract [Mostowski 1996] for some more information.

4 Henkin quantifiers with dualization and computational complexity

This chapter concentrates on the interrelation between our results and descriptive complexity theory. Descriptive complexity theory is a systematical attempt to provide descriptive (i.e. logical) counterparts for notions known from computational complexity theory. For a very brief account of the main ideas of DCT, see the Appendix; for a comprehensive treatment, we refer to [Immerman 1998] or to [Ebbinghaus–Flum 1995].

There are some interesting theorems showing that certain sublogics of the logic with Henkin quantifiers capture important computational complexity classes. We give two key examples. Firstly, it is an easy corollary of Fagin's theorem (theorem A.12) and corollary 2.17 that the class of all simple formulae with (standard) Henkin quantifiers captures **NP**.

Theorem 4.1 ([Blass–Gurevich 1986]) *$N^+(\mathcal{H})$ captures* **NP**.

Secondly, let a Blass–Gurevich quantifier[14] be a quantifier of the form

$$\begin{array}{l} \forall x_1 \ \ldots \ \forall x_n \ \exists \alpha \\ \forall z_1 \ \ldots \ \forall z_k \ \exists \beta, \end{array}$$

where n, k are natural numbers and α, β are boolean variables — in other words, variables which may assume only two possible values. We write $\mathcal{BG}$ to denote the set of all Blass–Gurevich quantifiers. Blass and Gurevich proved the following theorem.

Theorem 4.2 ([Blass–Gurevich 1986]) *$N^+(\mathcal{BG})$ captures* **NL** *over ordered models.*

It is perhaps worth noting that the original theorem of [Blass–Gurevich 1986] was that $N^+(\mathcal{BG})$ captures **co-NL**, the class of problems whose complements are in **NL**. We now know that this is equivalent to theorem 4.2, since **NL** = **co-NL**.

The connection of our results from Chapter 3 with DCT is twofold. On the one hand, our results imply new theorems similar to the ones above. These theorems provide us with new examples of natural logics capturing important complexity classes between **NP** and **PSPACE**. On the other hand, thanks to results form DCT, we are able to gain new insight into the behaviour of sublogics of LBD in finite ordered models.

Let us recall a well–known theorem of descriptive complexity theory ($\mathbf{\Sigma^P_n}$ are levels of the so–called polynomial hierarchy, cf. Appendix):

[14] In [Blass–Gurevich 1986], Blass–Gurevich quantifiers were called narrow quantifiers.

Theorem 4.3 ([Stockmeyer 1977]) *For any $n \in \omega - \{0\}$, Σ_n^1 captures $\mathbf{\Sigma_n^P}$.*

Via this result, we have an immediate but interesting corollary of theorem 3.10:

Corollary 4.4 *For any $n \in \omega$, $N^+(PREF_n^d)$ captures $\mathbf{\Sigma_{n+1}^P}$.*

Another corollary follows easily from a theorem proved by Gottlob. To state the theorem, we will need to define some additional notions.

Definition 4.5 *A problem L over alphabet A is* **NP***–reducible to a problem L' if there exists an* **NP***–transducer M such that for each $w \in A^*$, $w \in L$ iff $out_M(w) \cap L' \neq \emptyset$, where $out_M(w)$ consists of all outputs of the transducer M on input w. A complexity class* **C** *is closed under* **NP***–reductions if every problem which is* **NP***–reducible to a problem in* **C** *is itself in* **C**.

Definition 4.6 *A complexity class* **C** *is closed under conjunctions if for any L in* **C**, *also $(L\#)^*L$ (the set of finite, non-empty strings of words from L separated by some symbol $\#$ not in the alphabet of L) is in* **C**[15].

Definition 4.7 *The marked union $L \oplus L'$ of two problems L and L' is the problem $0L \cup 1L'$. A complexity class* **C** *is closed under marked union if for all L, L' in* **C**, *also $L \oplus L'$ is in* **C**.

Theorem 4.8 ([Gottlob 1997]) *Let* **C** *be a complexity class closed under* **NP***–reductions, under conjunctions, and under marked union. If $\mathcal{Q}$ is a set of quantifiers such that $N^+(\mathcal{Q})$ captures* **C** *over ordered models, then $FO(\mathcal{Q})$*[16] *captures $\mathbf{L^C}$ over ordered models.*

Corollary 4.9 *For any $n \in \omega$, LBD_n captures $\mathbf{L^{\Sigma_{n+1}^P}}$ over ordered models.*

Proof. LBD_n is $FO(PREF_n^d)$ by definition, so we only have to check that the $\mathbf{\Sigma_n^P}$ classes satisfy the required closure conditions. This is a known fact, but we give a brief justification nonetheless.

NP*–reductions.* Let L be a problem which is **NP**–reducible (using a transducer T) to a problem L' in $\mathbf{\Sigma_n^P}$. Let M be a $\mathbf{\Sigma_n^P}$–machine for L'. A $\mathbf{\Sigma_n^P}$–machine $\tilde{M}$ for L may be constructed as follows. On input word w, $\tilde{M}$

[15]Formally, if we treat complexity classes as functors (see the Appendix), we should say that **C** is closed under conjunctions if for any alphabet A, any $L \in \mathbf{C}_A$, and any symbol $\# \notin A$, we have $(L\#)^*L \in \mathbf{C}_{A \cup \{\#\}}$.

[16]The paper [Gottlob 1997] uses the notation $\mathcal{Q}^+$ for $N^+(\mathcal{Q})$. $FO(\mathcal{Q})$ is first order logic extended by adding all quantifiers from $\mathcal{Q}$, for example $FO(PREF)$ is L^r.

simulates T — using, for example, an extra work tape instead of the output tape of T — until an accepting state is reached. Let v be the word written on the extra work tape at that point. $\tilde{M}$ now simulates M on v and accepts w whenever M accepts v.

Conjunctions. Let M be a $\mathbf{\Sigma_n^P}$–machine for some problem L. We can modify M to obtain a $\mathbf{\Sigma_n^P}$–machine $\tilde{M}$ for $(L\#)^*L$, where $\#$ is some symbol not in the alphabet of L. Consider a word w over the alphabet obtained from the alphabet of L by adding $\#$. $\tilde{M}$ first checks whether w is of the form $w_1\# \ldots \#w_k$, where the w_i's are words over the alphabet of L. If not, $\tilde{M}$ rejects. Otherwise, $\tilde{M}$ starts to simulate M on w_1. Then, once some computation reaches an accepting state, $\tilde{M}$ begins to simulate M on w_2, and so on, until all of the w_i's have been accepted. Then — and only then — $\tilde{M}$ accepts.

Marked union. This proof is by induction on n. We skip the base step as it is only a simplified version of the inductive step. Assume that $\mathbf{\Sigma_n^P}$ is closed on marked union and consider L and L' in $\mathbf{\Sigma_{n+1}^P}$. Let M, M' be $\mathbf{\Sigma_{n+1}^P}$–machines for L and L', respectively (using $\mathbf{\Sigma_n^P}$ oracles S and S', respectively). The $\mathbf{\Sigma_{n+1}^P}$–machine $\tilde{M}$ for $L \oplus L'$ uses oracle $S \oplus S'$ (which is in $\mathbf{\Sigma_n^P}$ by the inductive assumption). Let $w = \alpha v$ be an input for $\tilde{M}$ (where α is 0 or 1). $\tilde{M}$ first checks whether α is 0 or 1. Assume $\alpha = 0$ (the case for $\alpha = 1$ is analogous). Then $\tilde{M}$ operates exactly as M on v, the sole exception being that all oracle queries must be preceded by a 0. □

Gottlob also proved a theorem which provides us with a certain normal form for LBD_n (and hence, LBD) in finite ordered models.

Definition 4.10 *For a set of quantifiers $\mathcal{Q}$, an $FO(\mathcal{Q})$–formula is in Stewart Normal Form (SNF) if it is of the form*

$$\exists \bar{y}[(Q\bar{x}\psi(\bar{x}, \bar{y})) \wedge (\neg Q'\bar{x}'\psi'(\bar{x}', \bar{y}))],$$

where $Q, Q' \in \mathcal{Q}$ and ψ, ψ' are first order formulae.

Theorem 4.11 ([Gottlob 1997]) *Let $\mathcal{Q}$ be a set of generalized quantifiers such that $N^+(\mathcal{Q})$ captures the complexity class $\mathbf{C}$ over ordered models. If $\mathbf{C}$ is closed under $\mathbf{NP}$–reductions, under conjunctions, and under marked union, then for each $FO(\mathcal{Q})$–formula φ there exists an $FO(\mathcal{Q})$–formula ψ in SNF such that φ and ψ are equivalent in finite ordered models.*

We immediately obtain:

Corollary 4.12 *If φ is an LBD_n–formula, then there exists an LBD_n–formula ψ in SNF such that φ and ψ are equivalent in finite ordered models.*

Obviously, as every LBD–formula is an LBD_n–formula for some n, we also have that for every LBD–formula there exists a formula in SNF equivalent to it in finite ordered models. Moreover, the dualization depth of the quantifiers in the SNF–formula does not exceed the maximum of the dualization depths of quantifiers in the original formula.

Definition 4.13 *For a class of formulae $\mathcal{C}$, $Bool(\mathcal{C})$ is the closure of $\mathcal{C}$ on boolean combinations; $QBool(\mathcal{C})$ is the closure of $\mathcal{C}$ on boolean combinations and first order quantification.*

Theorem 4.11 implies that over finite ordered models, $LBD_n \equiv QBool(\Sigma^1_{n+1})$ (and hence, that $QBool(\Sigma^1_n)$ captures $\mathbf{L}^{\mathbf{\Sigma^P_n}}$ over ordered models). Indeed, by corollary 4.12 and theorem 3.10 every LBD_n–formula is equivalent in finite ordered models to a formula in $QBool(\Sigma^1_{n+1})$. Also by theorem 3.10, it is obvious that $LBD_n \geq QBool(\Sigma^1_{n+1})$ holds in general.

Over infinite models, the inclusions

$$\Sigma^1_n \subseteq Bool(\Sigma^1_n) \subseteq QBool(\Sigma^1_n) \subseteq \Delta^1_{n+1}$$

are all strict. Nothing is known about the strictness of these inclusions over finite ordered nor even simply finite models. It is widely believed that some if not all of the inclusions are strict. However, this could be difficult to prove, since the strictness of any of the inclusions would imply $\mathbf{P} \neq \mathbf{NP}$.

A Complexity theory

In order to make the thesis reasonably self–contained, we list a few key definitions, ideas, and facts from complexity theory and descriptive complexity theory (DCT). For more on complexity theory, see [Balcázar et al. 1995]; for more on DCT, see [Immerman 1998] or [Ebbinghaus–Flum 1995].

Definition A.1 *A (nondeterministic) Turing machine (TM) is a tuple* $M = (A, Q, q_s, q_A, \delta)$*, where:*

A is a finite non–empty set (the alphabet of M; we shall normally assume that A contains at least the symbols 0 *and* 1*),*

Q is a finite non–empty set (the set of states of M),

q_s, q_A are distinguished states (initial and accepting, respectively), and

$\delta : A^2 \times Q \longrightarrow P(A \times Q \times \{L, R, N\}^2)$ is a partial function known as the transition function of M.

M is a deterministic Turing machine (DTM) if for all $a_1, a_2 \in A, q \in Q$ the set $\delta(a_1, a_2, q)$ contains at most one element. Otherwise M is essentially nondeterministic.

A TM is thought of as a device consisting of two semi–infinite tapes, the work tape and the input tape, both divided into cells (with a leftmost but no rightmost cell)[17]. Each of the tapes is equipped with a tape head which can move right or left, scanning the cells of the tape, one at a time.

TMs operate on words over A, i.e. finite strings of symbols from A (the set of words over A is denoted by A^*). A machine operating on an input word w starts in the initial state with the input tape containing w followed by a fixed symbol "blank" in all other cells, and all cells of the work tape also containing "blank". The heads initially scan the leftmost cells of the tapes. At any given moment, the machine is in one of the states. It then reads the contents of the scanned cells on both tapes. Now, it may change the contents of the work tape cell by writing a new alphabet symbol on it (the input tape is read–only), move each of the heads right or left to scan a new cell, and change the state. Together, these operations form a step. What the machine actually does in a given step depends on its current state and the contents of the scanned cells — via the transition function. If the transition function for a given state and cell contents is undefined, the machine stops.

Definition A.2 *Given a machine M, a configuration of M is a triple (q, a_1, a_2), where q is the current state of M, a_1 represents the contents of*

[17] We may allow Turing machines to have more than one work tape. The formal definition must then be appropriately modified.

the input tape, and a_2 represents the contents of the work tape. The a_i's are elements of $A^*\#A^{*}$[18], where # is some symbol not in A, used to denote the position of the tape head (the convention is that the head scans the symbol immediately to the right of #). All symbols on the tape not appearing in the appropriate a_i are assumed to be the symbol "blank".

A configuration is initial if it contains the initial state and the heads scan the leftmost symbols of the cells; it is accepting if it contains the accepting state. A configuration c_1 is a successor of c_0 if the transition function δ allows the transition from c_0 to c_1 to occur in one step.

A computation of M on input $w \in A^*$ is a finite sequence $c_0, \ldots, c_k$ of configurations of M such that c_0 is an initial configuration with w on the input tape, c_i is a successor of c_{i-1} (for $i = 1, \ldots, k$), and c_k is a configuration for which the transition function is undefined. An accepting computation is one which ends in an accepting configuration. The length of a computation is the number $k + 1$.

From now on, we consider only such machines M that for each word w, there is at least one computation of M on w.

Proposition A.3 *If M is a DTM and $w \in A^*$, then there is exactly one computation of M on w.*

A problem (or language) L over alphabet A is any subset of A^*. For a given machine M we define $L(M)$ — the problem recognized by M — as the set of those words w over A for which there is an accepting computation of M on input w.

We may now define notions of running time and work space of a TM. Note that both these notions have two versions, deterministic and general (nondeterministic); the former make sense only for deterministic machines. Note also that the length of a word w is the number of symbols in w.

Definition A.4 *Let M be a Turing machine.*

- *if M is a DTM and w is a word, then the deterministic computation time of M on w is the length of the computation of M on input w.*
- *if M is a DTM, its deterministic running time is the partial function defined by: $dtime_M(n)$ = the maximum deterministic computation time of M on an input of length n.*

[18]There is a convention that if L and L' are problems, then LL' is the problem consisting of words which are concatenations of a word from L and a word in L', in that order. Furthermore, when L is a one–word problem $\{w\}$, we write wL' instead of $\{w\}L'$.

- *if w is a word, then the computation time of M on w is the length of the shortest accepting computation of M on w, if $w \in L(M)$; otherwise it is* 1.
- *the running time of M is the function defined by: $time_M(n)$ = the maximum computation time of M on an input of length n.*
- *if M is a DTM and w is a word, then the deterministic computation space of M on w is the number of work tape cells scanned during the computation of M on w.*
- *if M is a DTM, its deterministic work space is the partial function defined by: $dspace_M(n)$ = the maximum deterministic computation space of M on an input of length n.*
- *if w is a word, then the computation space of M on w is the minimal number of work tape cells scanned during an accepting computation of M on w, if $w \in L(M)$; otherwise it is 1.*
- *the work space of M is the function defined by: $space_M(n)$ = the maximum computation space of M on an input of length n.*

It is possible to define classes of problems which can be recognized within resources limited by some given function:

Definition A.5 *Let $f : \omega \longrightarrow \omega$ and $g : \omega \longrightarrow \omega$ be total functions. Then $g = O(f)$[19] if for some natural number $r > 0$, $g(n) < r \cdot f(n)$ for all but finitely many n.*

Definition A.6 *Let $L \subseteq A^*$ and let $f : \omega \longrightarrow \omega$ be a total function.*

- *$L \in DTIME(f)$ if and only if there exists a DTM M such that $L(M) = L$ and $dtime_M = O(f)$;*
- *$L \in NTIME(f)$ if and only if there exists a TM M such that $L(M) = L$ and $time_M = O(f)$;*
- *$L \in DSPACE(f)$ if and only if there exists a DTM M such that $L(M) = L$ and $dspace_M = O(f)$;*
- *$L \in NSPACE(f)$ if and only if there exists a TM M such that $L(M) = L$ and $space_M = O(f)$.*

[19]The notation $g = O(f)$ is slightly awkward, but it is the most common in the literature. $O(f)$ is a set of functions somehow related to f, which would suggest the notation $g \in O(f)$.

Complexity classes are, in general, just classes of problems computable within some limit of resources. Formally, we treat a complexity class as a functor $\mathbf{C}$ which assigns to every alphabet A the appropriate set $\mathbf{C}_A$ of problems over A.

The following list contains definitions of some of the most important complexity classes.

Definition A.7 (basic complexity classes)

- $\mathbf{L} = DTIME(log_2 n)$;
- $\mathbf{NL} = NTIME(log_2 n)$;
- $\mathbf{P} = \bigcup_{k\in\omega} DTIME(n^k)$;
- $\mathbf{NP} = \bigcup_{k\in\omega} NTIME(n^k)$;
- $\mathbf{PSPACE} = \bigcup_{k\in\omega} DSPACE(n^k)$;
- $\mathbf{EXPTIME} = \bigcup_{k\in\omega} DTIME(2^{n^k})$.

The above classes are listed in order of inclusion. It is suspected that all of these inclusions are strict, although only the inclusions $\mathbf{NL} \subseteq \mathbf{PSPACE}$ and $\mathbf{P} \subseteq \mathbf{EXPTIME}$ are known to be strict. Proving (or disproving) the strictness of the remaining inclusions is one of the main tasks of complexity theory.

We need to outline two more important concepts of complexity theory, oracles and many–one reductions. To use oracles, we introduce a new type of TMs — oracle machines — with an additional tape called the oracle tape (the definition of the transition function is modified accordingly). The oracle tape is write–only. During a computation, an oracle machine may enter a special state called the query state. Entering the query state is interpreted as asking a (truthtul and competent) oracle whether the word currently written on the oracle tape belongs to some fixed problem S (which is called the oracle set and may be any subset of A^*, even an uncomputable one, i.e. one which is not recognized by any TM). Once the query is made, in unit time the query tape is erased and the machine enters a special "YES" or "NO" state according to whether the word which was written on the oracle tape belongs to S or not. The computation then continues.

Obviously, the problem recognized by an oracle machine depends in general not only on the machine itself, but also on the oracle set. We use the notation $L(\{M\}^S)$ to indicate the set recognized by M with oracle set S.

By analogy to the non–oracle case, we may speak of oracle complexity classes. If $\mathbf{C}$, $\mathbf{D}$ are complexity classes, then a problem L is in the class $\mathbf{C^D}$ if $L = L(\{M\}^S)$ for some $\mathbf{C}$–oracle machine M and oracle set S belonging to the class $\mathbf{D}$[20]. In the case where $\mathbf{C}$ is space–bounded, it is sometimes — but not always — assumed that the oracle tape is also subject to the space bound; this is a subtle issue, and we will not go into details.

In chapter 4, we discussed some results concerning the so–called polynomial hierarchy. This hierarchy consists of oracle complexity classes.

Definition A.8 *The definition of the polynomial hierarchy runs inductively as follows:*

$$\mathbf{\Sigma_1^P} = \mathbf{NP},$$

$$\mathbf{\Sigma_{n+1}^P} = \mathbf{NP}^{\mathbf{\Sigma_n^P}},$$

$$\mathbf{\Pi_n^P} = \mathbf{co}{-}\mathbf{\Sigma_n^P},$$

where $\mathbf{co}{-}\mathbf{C}$ *is the class of complements of problems in* $\mathbf{C}$ *(relative to appropriate alphabets).*

Almost all of these classes ($\mathbf{\Pi_1^P}$ being the possible exception) contain $\mathbf{NP}$ and are contained in $\mathbf{PSPACE}$.

To talk about many–one reductions, we introduce yet another type of TMs, namely output–computing machines or *transducers*. Transducers are TMs with an extra write–only tape called the output tape. A transducer M computes a function whose domain is $L(M)$ and whose values are, in general, subsets of A^*. For a word w in $L(M)$, the set of values of this function for w (denoted by $out_M(w)$) consists of those words which may appear on the output tape when a computation of M on input w ends in an accepting state. Note that the functions computed by deterministic transducers always have only singletons as values and thus may be treated as functions into A^*; we use the notation $\{M\}(w)$ in this case.

The basic idea of many–one reductions is the following. Given two problems L and L', assume that there exists a transducer M such that for every word w, w is in L if and only if at least one of the elements of $out_M(w)$ is

[20] Actually, this "definition" is not fully correct, because the complexity class $\mathbf{C}$ does not — treated extensionally, as a class of problems — impose any bounds on the resources used by M. These bounds are only imposed once we specify a family of functions which is to define the class $\mathbf{C}$ and the resources which are to be limited by this family. In general, the same complexity class $\mathbf{C}$ (understood extensionally) may be defined by different resource bounds, which can lead to a situation in which essentially different classes will all satisfy our "definition" of $\mathbf{C^D}$. Thus, the "definition" should be treated only as a notational convention the sense of which should always be determined by a specific context.

in L'. We may then say that M — in some sense — reduces the problem L to the problem L'. This is perhaps most evident when M is deterministic. It is natural, and indeed necessary for the idea to be of any interest, to impose some complexity restrictions on transducers performing many–one reductions (running time and work space are defined for transucers similarly as for standard TMs). Such restrictions give rise to reductions defined in terms of complexity classes and, in turn, to complexity classes defined in terms of such reductions. This is hard to discuss on a general level, so we define only a few important examples.

Definition A.9 *A problem L over alphabet A is:*

- *polynomial–time (logspace) reducible to a problem L' if there exists a deterministic transducer M working in polynomial time (logarithimic space) such that for each $w \in A^*$, $w \in L$ if and only if $\{M\}(w) \in L'$.*
- **NP***–reducible to a problem L' if there exists an* **NP***–transducer M such that for each $w \in A^*$, $w \in L$ if and only if $out_M(w) \cap L' \neq \emptyset$.*
- **NL***–reducible to a problem L' if there exists a polynomial time* **NL***–transducer M such that for each $w \in A^*$, $w \in L$ if and only if $out_M(w) \cap L' \neq \emptyset$.*

If **C**, **D** are complexity classes and if **D**–many–one reductions be defined[21], then a problem $L \in \mathbf{C}$ is said to be **C**–complete via **D**–reductions if and only if every problem in **C** is **D**–reducible to L. The study of problems which are **NP**–complete (via polynomial–time reductions) has been especially intensive. Today we know of hundreds of such problems. Historically, the first problem discovered to be **NP**–complete was the problem SAT.

Theorem A.10 ([Cook 1971]) *Let SAT be the problem defined as follows: given a propositional formula φ, $\varphi \in SAT$ if and only if φ is satisfiable. SAT is* **NP***–complete via polynomial–time reductions.*

We now turn to descriptive complexity theory. The main idea behind this theory is that finite models of a fixed vocabulary (from now on: "model" means "finite model") may be identified with their descriptions (i.e. words over some alphabet, for example $\{0, 1\}$) via some reasonable coding. In this way, questions about the computational complexity of problems may be

[21]Similarly to the definition of $\mathbf{C}^{\mathbf{D}}$ and for similar reasons (see the previous footnote), the definition of **D**-reductions cannot depend on the class **D** alone, but rather on a given class of functions defining the class **D** and on some additional factors. Hence, what is meant by **D**-reductions should always be specified in any given context.

translated into questions about the descriptive complexity of the corresponding classes of models (that is, about how much expressive force we need to describe these classes).

To make this a little more precise, assume τ is some fixed relational vocabulary, possibly with constants — e.g. $\tau = (R_1, \ldots, R_k, c_1, \ldots, c_s)$, with each R_i r_i–ary. We additionally assume that models of τ are ordered in some standard way, for example that their universes are initial segments of the natural numbers and that τ contains an extra binary relation symbol S which is *not* listed among the R_i's and is always interpreted as the appropriate restriction of the standard successor relation on the integers.

We then identify a model M with a word over $\{0, 1\}$ which is a concatenation of the following. Firstly, a series of as many ones as there are elements of $|M|$, followed by a 0. Secondly, for each R_i, a word of length $|M|^{r_i}$ whose l–th letter is 1 if and only if the l–th (lexicographically) tuple of elements of M is in R_i^M. Finally, for each c_j, the standard $log(|M|)$–long binary code of c_j^M.

A property of models of a given vocabulary τ is simply any class of such models. For a complexity class $\mathbf{C}$, a property is $\mathbf{C}$–decidable if it belongs to $\mathbf{C}$ when treated as a problem over $\{0, 1\}$. A sentence of some logic expresses a property if this property is exactly the class of all models in which the sentence is true.

Definition A.11 *A logic $\mathcal{L}$ captures the complexity class $\mathbf{C}$ over ordered models if the following two conditions are satisfied:*

(i) each $\mathbf{C}$–decidable property over finite ordered models (in any given vocabulary) can be expressed by an $\mathcal{L}$–sentence;

(ii) for each $\mathcal{L}$–sentence φ in vocabulary τ, the problem $Mod_\tau(\varphi)$ (restricted to finite models) is in $\mathbf{C}$.

Sometimes, the condition that we restrict our attention to ordered models may be dropped. In that case, we may code finite models of an arbitrary relational vocabulary σ as words over $\{0, 1\}$ — just as previously — but we consider only classes of models closed on isomorphisms (and problems corresponding to such classes). Otherwise, for example, some problems which are very easy from the computational point of view (even problems containing only one word) would correspond to classes of models not expressible in any logic (the reason being that they are not closed on isomorphisms).

This approach leads to a general notion of a logic capturing a complexity class (over arbitrary models), defined analogously to capturing over ordered models, with the restriction mentioned above. The question: "When does

a logic which captures a complexity class over ordered models capture this class also in the general sense?" belongs to the most difficult and important questions of descriptive complexity theory. It is known that many logics which capture some complexity classes over ordered models fail to do so over arbitrary models [22].

Thanks to the concept of logics capturing complexity classes, questions regarding e.g. the strictness of inclusions between complexity classes can now be treated as questions about the semantical power of various logics. We limit ourselves to just one example of a theorem on a logic capturing a complexity class (some more examples are given in chapter 4). This theorem was the starting point of descriptive complexity theory.

Theorem A.12 ([Fagin 1974]) Σ_1^1 *captures* **NP** *(over arbitrary (finite) models).*

[22]For a discussion of the role of ordering in DCT, we refer to Chapter 12 of [Immerman 1998] or to appropriate fragments of Chapters 6 and 10 of [Ebbinghaus–Flum 1995].

References

[Balcázar et al. 1995] J. L. Balcázar, J. Díaz, and J. Gabarró, **Structural Complexity I. Second, Revised Edition**, Springer-Verlag 1995.

[Blass–Gurevich 1986] A. Blass and Y. Gurevich, *Henkin Quantifiers and Complete Problems*, in **Annals of Pure and Applied Logic** 32(1986), pp. 1–16.

[Chang–Keisler 1990] C. C. Chang and H. J. Keisler, **Model Theory**, North Holland Publishing Company 1990.

[Cook 1971] S. Cook, *The Complexity of Theorem-Proving Procedures*, in **Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing**, 1971, pp. 151–158.

[Ebbinghaus–Flum 1995] H.-D. Ebbinghaus and J. Flum, **Finite Model Theory**, Springer-Verlag 1995.

[Enderton 1970] H. B. Enderton, *Finite Partially–Ordered Quantifiers*, in **Zeitschrift für Mathematische Logik und Grundlagen der Mathematik** 16(1970), pp. 393–397.

[Fagin 1974] R. Fagin, *Generalized First-Order Spectra and Polynomial-Time Recognizable Sets*, in R. Karp (ed.) **Complexity of Computation, SIAM-AMS Proceedings** 7(1974), pp. 43–73.

[Gottlob 1997] G. Gottlob, *Relativized Logspace and Generalized Quantifiers over Finite Ordered Structures*, in **The Journal of Symbolic Logic** 62(1997), pp. 545–574.

[Harel 1979] D. Harel, *Characterizing Second Order Logic with First Order Quantifiers*, in **Zeitschrift für Mathematische Logik und Grundlagen der Mathematik** 25(1979), pp. 419–422.

[Henkin 1961] L. Henkin, *Some Remarks on Infinitely Long Formulae*, in **Infinitistic Method** Pergamon Press and PWN 1961, pp. 167–183.

[Hintikka 1974] J. Hintikka, *Quantifiers vs. Quantification Theory*, in **Dialectica** 27(1974), pp. 329–358.

[Immerman 1998] N. Immerman **Descriptive Complexity**, Springer-Verlag 1998.

[Krynicki–Mostowski 1995] M. Krynicki and M. Mostowski *Quantifiers, Some Problems and Ideas*, in M. Krynicki, M. Mostowski, and L.W. Szczerba (eds.) **Quantifiers: Logics, Models and Computation** Volume I, Kluwer Academic Publishers, 1995, pp. 1–19.

[Krynicki–Mostowski 1995a] M. Krynicki and M. Mostowski *Henkin Quantifiers*, in M. Krynicki, M. Mostowski, and L.W. Szczerba (eds.) **Quantifiers: Logics, Models and Computation** Volume I, Kluwer Academic Publishers, 1995, pp. 193–262.

[Mostowski 1987] M. Mostowski, *The Relational Semantics for Branched Quantifiers*, in D. Skordev (ed.) **Mathematical Logic and its Application**, Plenum Press, 1987, pp. 315–322.

[Mostowski 1987a] M. Mostowski, *Branched Quantifiers with Dualization*, technical report for Research project RPBP III. 24, Białystok 1987.

[Mostowski 1987b] M. Mostowski, *An Extension of the Logic with Branched Quantifiers*, in **8th International Congress of Logic, Methodology, and Philosophy of Science, Abstracts**, vol. 5, part 3, Moscow 1987, pp.261–263.

[Mostowski 1991] M. Mostowski, **Branched Quantifiers**, Rozprawy Uniwersytetu Warszawskiego, Dział Wydawnictw Filii UW w Białymstoku, 1991.

[Mostowski 1991a] M. Mostowski, *Arithmetic with the Henkin quantifier and its Generalizations*, in F. Gaillard, D. Richard (eds.) **Seminaire du Laboratoire Logique, Algorithmique et Informatique Clermontois**, Volume II (1989–1990), 1991, pp. 1–25.

[Mostowski 1995] M. Mostowski *Quantifiers Definable by Second Order Means*, in M. Krynicki, M. Mostowski, and L.W. Szczerba (eds.) **Quantifiers: Logics, Models and Computation** Volume II, Kluwer Academic Publishers, 1995, pp. 181–214.

[Mostowski 1996] M. Mostowski, *Hierarchies Determined by Henkin and Finite-Order Quantification*, in **Bulletin of Symbolic Logic** 2(1996), pp. 228-229.

[Mostowski–Zdanowski 2002] M. Mostowski and K. Zdanowski, *Henkin Quantifiers in Finite Models*, in preparation.

[Stockmeyer 1977] L. Stockmeyer *The Polynomial-Time Hierarchy*, in **Theoretical Computer Science** 3(1977), pp. 1–22.

[Walkoe 1970] W.J. Walkoe, *Finite partially-ordered quantification*, in **Journal of Symbolic Logic** 35(1970), pp. 535–555.