On the Complexity of Horn and Krom Fragments of Second-Order Boolean Logic

Miika Hannula
Department of Mathematics and Statistics, University of Helsinki, Finland
miika.hannula@helsinki.fi

Juha Kontinen
Department of Mathematics and Statistics, University of Helsinki, Finland
juha.kontinen@helsinki.fi

Martin Lück
Institut für Theoretische Informatik, Leibniz Universität Hannover, Germany
lueck@thi.uni-hannover.de

Jonni Virtema
Faculty of Humanities and Human Sciences, Hokkaido University, Sapporo, Japan
jonni.virtema@let.hokudai.ac.jp

Abstract

Second-order Boolean logic is a generalization of QBF, whose constant alternation fragments are known to be complete for the levels of the exponential time hierarchy. We consider two types of restriction of this logic: 1) restrictions to term constructions, 2) restrictions to the form of the Boolean matrix. Of the first sort, we consider two kinds of restrictions: firstly, disallowing nested use of proper function variables, and secondly stipulating that each function variable must appear with a fixed sequence of arguments. Of the second sort, we consider Horn, Krom, and core fragments of the Boolean matrix. We classify the complexity of logics obtained by combining these two types of restrictions. We show that, in most cases, logics with k alternating blocks of function quantifiers are complete for the kth or $(k-1)$th level of the exponential time hierarchy. Furthermore, we establish **NL**-completeness for the Krom and core fragments, when $k = 1$ and both restrictions of the first sort are in effect.

2012 ACM Subject Classification Theory of computation → Complexity theory and logic

Keywords and phrases quantified Boolean formulae, computational complexity, second-order logic, Horn and Krom fragment

Digital Object Identifier 10.4230/LIPIcs.CSL.2021.27

Funding The first and second authors were supported by grant 308712 of the Academy of Finland. The fourth author was an international research fellow of Japan Society for the Promotion of Science (Postdoctoral Fellowships for Research in Japan (Standard)).

Acknowledgements We wish to thank the anonymous referees for their very careful and thorough reviews.

1 Introduction

The canonical complete problem for **PSPACE** is the *quantified Boolean formula problem* (QBF) [17]. This generalization of the *Boolean satisfiability problem* (SAT) asks whether a Boolean sentence of the form $Q_1 p_1 \ldots Q_n p_n \psi$, where $Q_i \in \{\exists, \forall\}$, is true. Today QBF attracts widespread interest in diverse research communities. In particular, QBF solving techniques are important in application domains such as planning, program synthesis and verification, adversary games, and non-monotonic reasoning, to name a few [15]. A further generalization of QBF is the *dependency quantified Boolean formula problem* (DQBF) [13, 12].

This problem, complete for *nondeterministic exponential time* (**NEXP**), asks whether a Boolean sentence of the form

$$\forall p_1 \dots \forall p_n \exists q_1 \dots \exists q_m \psi$$

with constraints $C_i \subseteq \{p_1, \dots, p_n\}$ is true; here, the selection of truth values for q_i may only depend on that of those variables that are in C_i. In other words, DQBF enriches QBF by allowing nonlinear dependency patterns between variables. DQBF-specifications can be exponentially more succinct compared to that of QBF and have found applications in areas such as non-cooperative games, SMT, and bit-vector logics. Furthermore, the development of DQBF-solvers is also well under way [14].

Put in different terms, DQBF instances can be seen as Boolean sentences of the form

$$\exists f_1 \dots \exists f_m \forall p_1 \dots \forall p_n \psi,$$

where each f_i is a Boolean function variable whose occurrences in ψ are of the form $f_i(p_{i_1}, \dots, p_{i_k})$, for some fixed sequence of proposition variables $p_{i_1}, \dots, p_{i_k}$. In previous studies, extensions of DQBF with alternating function quantification have also been considered. The so-called *alternating dependency quantified Boolean formula problem* (ADQBF) was shown to be complete for alternating exponential time with polynomially many alternations (**AEXP**(poly)) in [6]. This work was preceded by the works of Lück [9] and Lohrey [8] studying second-order Boolean logic with explicit quantification of Boolean functions (denoted SO_2 in this work). Their results showed, e.g., that restricting the alternations of function quantification to $k-1$ yields complete problems for the kth levels of the exponential hierarchy.

In this article we embark on a systematic study of the complexity of fragments of SO_2, defined by combining restrictions on the structure of function terms and the Boolean matrix. A remarkable fact is that, when restricting attention to Horn formulae, all the complexity distinctions between SAT, QBF, and DQBF disappear. Bubeck and Büning [4] showed that those DQBF instances whose quantifier-free part is a conjunction of Horn clauses are solvable in polynomial time. Consequently, all the aforementioned problems over Horn formulae are **P**-complete. This implies that the high complexity of (D)QBF is not a straightforward consequence of its quantification structure; rather, structural complexity from the quantifier-free part is also needed. An immediate question is: How complex quantification is required to neutralize structural limitations, such as the Horn form, on the quantifier-free part? It is exactly this interplay between quantification and quantifier-free formula structure that will be the focus of this paper.

A formula of SO_2 is in Σ_k or in Π_k if it is in prenex normal form with $k-1$ alternations for function quantification, with the first quantifier block being respectively existential or universal. If the quantifier-free part of a formula is in conjunctive normal form, then it is called (a) *Horn* if each clause has at most one positive literal, (b) *Krom* if each clause contains at most two literals, and (c) *core* if it is both Horn and Krom. A formula is called (i) *simple* if it contains no nested function terms, and (ii) *unique* if in it each function variable is associated with a unique argument tuple. These last two criteria, in particular, are meaningful for formulae involving second-order quantification. Uniqueness and simpleness are also the characteristics of function terms introduced in the process of Skolemization, and more importantly, tacitly assumed in the DQBF problem. One of the goals of this paper is to determine the impact of such restrictions. This way we generalize the aforementioned results on DQBF, which can be understood in our terms as unique simple Σ_1.

Table 1 Complexity of fragments of second-order Boolean logic restricted to Horn, Krom, or core clauses. All entries are completeness results with respect to logspace-reductions. The $\star$ means "any". "H"and "$\in$" are used for references for the hardness and membership results respectively. All trivial upper bounds, i.e., of the form $\Sigma_k^{\mathrm{E}}/\Pi_k^{\mathrm{E}}$, are by Theorem 7.
†: Likely identical with first row. ‡: The result follows from some other result in the table.

Simpleness	Uniqueness	k	Clauses	Σ_k	Π_k	Reference	
Simple	Unique	$k = 1$	Horn	**P**	?	[4]	
			Krom/core	**NL**	**NL**	H/$\in$: 18	
		$k = 2$	Horn	Σ_2^{E}	?	H/$\in$: ‡	
			Krom/core	Σ_2^{E}	**NL**	H: 25, $\in$: ‡	H/$\in$: 18
		$k \geq 3$ odd	$\star$	$\Sigma_{k-1}^{\mathrm{E}}$	Π_k^{E}	H: 26 $\in$: ‡	H: 25 $\in$: ‡
		$k \geq 4$ even	$\star$	Σ_k^{E}	Π_{k-1}^{E}	H: 25 $\in$: ‡	H: 26 $\in$: ‡
	Non-unique	$k = 1$	Horn	**EXP**	Π_1^{E}	H/$\in$: 30	H/$\in$: ‡
			Krom/core	**PSPACE**	Π_1^{E}	H/$\in$: 29	H: 28, $\in$: ‡
		$k \geq 3$ odd	$\star$	$\Sigma_{k-1}^{\mathrm{E}}$	Π_k^{E}	H: ‡ , $\in$: 19	H/$\in$: ‡
		$k \geq 2$ even	$\star$	Σ_k^{E}	Π_{k-1}^{E}	H/$\in$: ‡	H: ‡, $\in$: 19
Non-simple	Unique	$k = 1$	Horn	Σ_1^{E}	?†	H/$\in$: ‡	
			Krom/core	Σ_1^{E}	**NL**	H: 23, $\in$: ‡	H/$\in$: 18
		$k \geq 2$	$\star$	Σ_k^{E}	Π_k^{E}	H: 23 H: 24, $\in$: ‡	
	Non-unique	$k \geq 1$	$\star$	Σ_k^{E}	Π_k^{E}	H: 23, 24, 28, $\in$: [8]	
$\star$	$\star$	$k = \omega$	$\star$	**AEXP**(poly)	**AEXP**(poly)	H: 25, $\in$: [6]	

Our contributions are the following. We show, on the one hand, that the complexity of DQBF over Krom or core formulae collapses to **NL**, and that this result extends to simple and unique Π_1 and Π_2. On the other hand, we show that almost all other cases are complete for the corresponding, or their neighboring, levels of the exponential hierarchy. Some cases are left open; most intriguing such case is the inverse of the DQBF-Horn problem (i.e., simple and unique Π_1 Horn), which is only known to be between **NL** and Π_1^{E}. A summary of our results can be found in Table 1.

2 Second-order quantified Boolean formulae

Second-order propositional logic is obtained from usual quantified Boolean formulae by shifting from quantification over proposition variables to quantification over Boolean functions. We call this logic SO_2, as it essentially corresponds to second-order predicate logic restricted to the domain $\{0, 1\}$.

2.1 Syntax and semantics

Let $\Phi = \{f_1, f_2, \ldots\}$ denote a countable set of function *variables*, each with an *arity* $\mathrm{ar}(f_i) \in \mathbb{N}$. We assume that there are infinitely many variables of any arity. Variables with arity 0 are called *propositional*. Variables with higher arity are called *proper function variables*. A *Φ-term* is either a propositional variable from Φ, or an expression of the form $f(t_1, \ldots, t_n)$, where $f \in \Phi$ is a variable of arity n and $t_1, \ldots, t_n$ are Φ-terms. The outermost variable in a term is called its *head*. The set of *subterms* $\mathsf{st}(t)$ of a term $t = f(t_1, \ldots, t_n)$ is recursively defined as $\{t\} \cup \bigcup_{i=1}^{n} \mathsf{st}(t_i)$. A term t appears *nested* in a term t' if $t \in \mathsf{st}(t') \setminus \{t'\}$. By identifying a term with its head, we also say that a *variable* appears nested in another term or variable. A *Φ-formula* is either a Φ-term, or an expression of the form $\varphi \wedge \varphi'$, $\neg\varphi$, or $\exists f \varphi$, where $f \in \Phi$ is a variable and φ, φ' are Φ-formulae. We write $\mathsf{SO}_2(\Phi)$ for the set of all Φ-formulae. We often omit Φ if it is clear from the context. The abbreviations $\forall f \varphi := \neg\exists f \neg\varphi$, $\varphi \vee \psi := \neg(\neg\varphi \wedge \neg\psi)$, $\varphi \to \psi := \neg\varphi \vee \psi$ and $\varphi \leftrightarrow \psi := (\varphi \to \psi) \wedge (\psi \to \varphi)$ are defined in the usual fashion. We sometimes make use of the logical constants 0 and 1, which can be expressed with quantified propositions that are forced to take the appropriate truth values. If $\vec{f} = (f_1, \ldots, f_n)$ is a tuple of variables, we sometimes write $\forall \vec{f}$ for $\forall f_1 \ldots \forall f_n$ and $\exists \vec{f}$ for $\exists f_1 \ldots \exists f_n$. Also, the formula $\vec{f} \leftrightarrow \vec{g}$, assuming $|\vec{f}| = |\vec{g}|$, is short for $\bigwedge_{i=1}^{|\vec{f}|}(f_i \leftrightarrow g_i)$.

We write $\mathrm{Var}(\varphi)$ ($\mathrm{Fr}(\varphi)$, resp.) to denote the set of variables that occur (occur freely, resp.) in φ. A formula with no free variables is *closed.* A term t is *free in* φ if $\mathrm{Var}(t) \subseteq \mathrm{Fr}(\varphi)$.

A *Φ-interpretation* I is a function that maps every variable $f \in \Phi$ to its interpretation $I(f)\colon \{0,1\}^{\mathrm{ar}(f)} \to \{0,1\}$. If I is a Φ-interpretation, $f \in \Phi$ has arity n, and $F\colon \{0,1\}^n \to \{0,1\}$, then I^f_F is the Φ-interpretation defined by $I^f_F(f) := F$ and $I^f_F(g) := I(g)$ for all $g \neq f$. The *valuation* $[\![\varphi]\!]_I \in \{0,1\}$ of a formula φ in I is defined as follows:

$$\begin{aligned}
[\![\varphi \wedge \psi]\!]_I &:= [\![\varphi]\!]_I \cdot [\![\psi]\!]_I,\\
[\![\neg\varphi]\!]_I &:= 1 - [\![\varphi]\!]_I,\\
[\![f(\varphi_1, \ldots, \varphi_n)]\!]_I &:= I(f)([\![\varphi_1]\!]_I, \ldots, [\![\varphi_n]\!]_I),\\
[\![\exists f \varphi]\!]_I &:= \max\left\{ [\![\varphi]\!]_{I^f_F} \;\middle|\; F : \{0,1\}^n \to \{0,1\} \right\}.
\end{aligned}$$

We often write $I \vDash \varphi$ instead of $[\![\varphi]\!]_I = 1$. We write $\varphi \vDash \psi$, if $I \vDash \varphi$ implies $I \vDash \psi$ for all suitable interpretations I. We say that φ and ψ are equivalent and write $\varphi \equiv \psi$, if $\varphi \vDash \psi$ and $\psi \vDash \varphi$. A Φ-formula φ is *valid* if $[\![\varphi]\!]_I = 1$ for all Φ-interpretations I. It is *satisfiable* if there is at least one I such that $[\![\varphi]\!]_I = 1$. Finally, a valid closed formula is called *true.*

2.2 Syntactic restrictions and normal forms

Next we consider basic normal forms of SO_2 such as *prenex form* and *conjunctive normal form.* These are defined as in classical QBF, except that a second-order literal may contain multiple variables in a nested way. Analogously to the classical case, we show that virtually all lower bounds already hold for those fragments. Here $[n]$ is used to denote the set of natural numbers $\{1, 2, \ldots, n\}$.

▶ **Definition 1.** *A* literal *is a term or the negation of a term. A* clause *is a disjunction of literals. A formula in* conjunctive normal form (CNF) *is a conjunction of clauses. A formula is a* Horn formula *if it is a CNF such that every clause contains at most one non-negated literal. A formula is a* Krom formula *if it is a CNF such that every clause contains at most two literals. A formula is a* core formula *if it is Horn and Krom.*

▶ **Definition 2** (Σ_k and Π_k)**.** *Let $k \geq 1$. The set Σ_k consists of all formulae of the form $Q_1\vec{f_1} \cdots Q_k\vec{f_k}\; Q_{k+1}\,\vec{x}\;\theta$, where $Q_i = \exists$ ($Q_i = \forall$) if i is odd (even), θ is quantifier-free, and $\vec{x}$ is a tuple of propositional variables. Moreover, we insist that all quantified variables are distinct. The analogous definition of Π_k is achieved by swapping $\exists$ and $\forall$.*

For unbounded quantifier prefixes, we write $\Sigma_\omega := \bigcup_{k\geq 1} \Sigma_k$ and $\Pi_\omega := \bigcup_{k\geq 1} \Pi_k$. A formula φ is in *prenex form* if it is in $\Sigma_\omega \cup \Pi_\omega$. A formula in prenex form is called Horn, Krom, or core, if its quantifier-free part is a CNF of the corresponding form.

Compared to classical QBF, the structure of second-order literals is much richer due to the ability to use nested Boolean functions, and because we can have function variables appear with different arguments. In this paper, we explore the complexity landscape that results from allowing second-order literals to occur only in a controlled fashion. In extension to the fragments introduced above, we define two classes of formulae that play major roles in the subsequent results: uniqueness and simpleness.

▶ **Definition 3** (Uniqueness). *A formula φ has* uniqueness *if for all pairs of terms of the form $f(t_1, \ldots, t_n)$ and $f(t'_1, \ldots, t'_n)$ that occur in φ, it holds that $t_i = t'_i$ for all $i \in [n]$.*

In other words, a function variable must always appear with the same arguments. For example, the formulae $f(0) \leftrightarrow f(1)$ and $\exists x \forall y (x \leftrightarrow f(y))$ both state that f is a constant function, but only the second one has uniqueness.

▶ **Definition 4** (Simpleness). *A formula is* simple *if functions occurring in it have only propositions as arguments.*

If a formula is not simple, it is not hard to restore simpleness by introducing additional existential variables. For example, $f(g(x))$ is equivalent to $\exists y\, (g(x) \leftrightarrow y \wedge f(y))$.

▶ **Proposition 5.** *For every SO_2-formula φ in prenex form there is a logspace-computable and simple formula ψ equivalent to φ.*

Proof. Suppose $\varphi = Q_1 f_1 \cdots Q_n f_n\, \theta$ with θ quantifier-free. Let $t_1, \ldots, t_k$ be an enumeration of all terms in θ. Then φ is equivalent to the formula

$$Q_1 f_1 \cdots Q_n f_n \exists y_1 \cdots \exists y_k \left(\theta^* \wedge \bigwedge_{i=1}^{k} (y_i \leftrightarrow t^*_i) \right),$$

where θ^* is obtained from θ by recursively replacing all terms t_j that occur nested inside other terms by y_j. ◀

▶ **Corollary 6.** *Let k be odd and let $\Psi \in \{\Pi_k, \Sigma_{k+1}\}$. Then for every formula $\varphi \in \Psi$ there is a logspace-computable formula $\psi \in \Psi$ that is simple and equivalent to φ. Furthermore, this translation preserves uniqueness, and the Horn, Krom and core property.*

If Ψ is a set of formulae, then Ψ^{s} is its restriction to simple formulae and Ψ^{u} is its restriction to formulae with uniqueness, and similarly Ψ^{h}, Ψ^{k} and Ψ^{c} for Horn, Krom, and core. E.g., Σ_2^{ush} is the set of all simple Σ_2-formulae with uniqueness which are in Horn CNF.

2.3 Known complexity results

We assume the reader to be familiar with basic complexity classes such as **PSPACE** and the exponential hierarchy, as well as logspace-reductions and basics of Turing machines. For a detailed exposition for these topics we refer the reader to [1] and to the complexity toolbox in Appendix A.

The quantifier alternation hierarchy of second-order Boolean logic is complete for the respective levels of the exponential time hierarchy, completely analogous to fragments of ordinary QBF being complete for the levels of the polynomial hierarchy.

▶ **Theorem 7** ([8, 9]). *Let $k \geq 1$. Truth of Σ_k-formulae is complete for Σ^{E}_k, and truth of Π_k-formulae is complete for Π^{E}_k.*

The result generalizes to unbounded number of quantifier alternations. The full logic is complete for the class **AEXP**(poly), that is, exponential runtime (corresponding to the size of second-order interpretations) but only polynomially many alternations (corresponding to the quantifier alternations in a formula with respect to its length).

▶ **Theorem 8** ([6, 9]). *Truth of $\Sigma_\omega^{\mathsf{us}}$-formulae, of Π_ω^{us}-formulae and of arbitrary* SO_2*-formulae is each complete for* $\mathbf{AEXP}(\text{poly})$.

However, as Bubeck and Büning [4] showed, the complexity even of second-order logic can drop down to tractable classes when the matrix (i.e., the quantifier free part) of the formula is restricted to Horn clauses:

▶ **Theorem 9** ([4]). *Truth of Σ_1^{ush}, that is, Σ_1-Horn formulae with simpleness and uniqueness, is* $\mathbf{P}$*-complete.*

2.4 Simplification based on variable dependencies

We conclude this section with a rather technical auxiliary result called *argument elision* that will be required in the subsequent sections. It allows to simplify formulae as follows. For example, the formula $\forall x\, \exists f\, \big(f(z,x) \leftrightarrow g(z)\big)$ can be simplified to an equivalent formula $\forall x\, \exists f_z\, \big(f_z(x) \leftrightarrow g(z)\big)$, for as the value of z is fixed to some $b \in \{0,1\}$ before f is quantified, the interpretations of f and f_z can be always copied from another such that $f_z(x)$ and $f(b,x)$ are the same functions. Hence the free variable z can be *elided* from the quantified function variable. Perhaps more relevant is the case where z is not free, but simply quantified before f. Indeed, the formulae $\forall z \forall x \exists f\, \big(f(z,x) \leftrightarrow g(z)\big)$ and $\forall z \forall x \exists f_z\, \big(f_z(x) \leftrightarrow g(z)\big)$ are equivalent.

Eliding the i-th position of a function variable f in a formula φ means to replace every quantifier Qf by Qg, where g is a fresh function variable of arity $\mathrm{ar}(f) - 1$ and $Q \in \{\exists, \forall\}$, and every term $f(t_1, \dots, t_n)$ with $g(t_1, \dots, t_{i-1}, t_{i+1}, \dots, t_n)$. If a formula has uniqueness (i.e., functions always appear with the same arguments $t_1, \dots, t_n$) then *eliding a term t* from a function variable f means the consecutive elision of all positions i such that $t_i = t$.

The following proposition follows via a simple inductive argument (see Appendix B).

▶ **Proposition 10** (Free term elision). *Let $\varphi \in \mathsf{SO}_2^{\mathsf{u}}$ be a prenex formula, f a function variable not free in φ, and t a term free in φ. Then eliding t from f yields a formula equivalent to φ.*

In particular, it follows that if $\varphi \in \Sigma_\omega^{\mathsf{u}}$ is a formula, f a function variable quantified in φ, and t a term such that all variables in $\mathrm{Var}(t)$ are quantified before f, then the elision of t from f produces an equivalent formula.

3 An NL-complete second-order fragment

In this section, we consider the Krom fragment and obtain tractability results for the first levels of the propositional second-order quantifier hierarchy. We show completeness for $\mathbf{NL}$, and hence obtain fragments that are as hard as the ordinary propositional Krom fragment. In our proofs, we follow the classical approach by Aspvall et al. [2], who showed that classical QBF with the quantifier-free part consisting of Krom clauses are solvable in $\mathbf{NL}$. The approach is to interpret the formula as an *implication graph* $G = (V, E)$. The crucial idea of the approach is that connectedness in the graph corresponds to logical implication. Here, V is the set of all literals in φ, closed under negation and $\neg\neg\ell$ identified with ℓ. An edge $(\ell_1, \ell_2) \in E$ exists when φ contains a clause equivalent to $\ell_1 \to \ell_2$, that is, of the form $\neg\ell_1 \vee \ell_2$. A unit clause ℓ is identified with $(\neg\ell \to \ell)$. A *strongly connected component* (or simply a *component*) S of G is a maximal subset of vertices such that for all distinct $v, v' \in S$ there is a path from v to v'. sec:lower-bounds

In classical propositional logic, a set of Krom clauses is satisfiable precisely if no cycle of the implication graph contains some literal ℓ and its negation $\neg\ell$ [2]. With quantifiers, the matter complicates and we need to account for the notion of *dependency* between variables. A literal t is called *universal* (*existential*) in φ if its head is quantified universally (existentially) in φ. A component is *universal* (*existential*) if it contains some (no) universal vertex.

A bit sloppily, we say that a literal ℓ is an *argument* of a literal ℓ' if there are $r \geq 1$, $i \in [r]$ and a term $f(t_1, \ldots, t_r)$ such that ℓ or $\neg\ell$ equals t_i, and ℓ' or $\neg\ell'$ equals $f(t_1, \ldots, t_r)$. In what follows, we restrict ourselves to simple fragments, that is, all arguments are propositions.

▶ **Definition 11.** *A vertex v* depends on *a vertex v', in symbols $v \rightsquigarrow v'$, if*

a) *v' is an argument of v, or*

b) *v' is quantified before v, and every argument of v' is either an argument of v or*

- *is quantified before v, if the argument is universal, and*
- *is quantified before or at the same quantifier block as v, if the argument is existential.*

If S and S' are components, we write $S \rightsquigarrow S'$ if some universal *vertex $u \in S$ depends on some vertex $v \in S'$ (with possibly $S = S'$).*

For classical Krom formulae, a QBF can be shown to be true if and only if the following conditions all hold [2]:

(1) There is no path from a universal vertex u to another universal vertex u' (with $u \neq u'$, but possibly $u = \neg u'$).

(2) No vertices v and $\neg v$ are in the same component.

(3) Every existential vertex v in the same component as a universal vertex u must depend on u.

Note: For classical QBF, (3) simply means that v must be quantified after u, but in the general case, we need the more complicated Definition 11. Moreover, we require another condition in addition to the above (1)–(3):

(4) There is no $\rightsquigarrow$-cycle among the components (including loops).

▶ **Example 12.** One formula that violates (4) is $\forall y_1 \forall y_2 \exists x_1 \exists x_2 (y_1(x_2) \leftrightarrow x_1) \wedge (y_2(x_1) \leftrightarrow x_2)$. The reason is that $y_1(x_2) \rightsquigarrow x_2$ and $y_2(x_1) \rightsquigarrow x_1$, and therefore $\{y_1(x_2), x_1\} \rightsquigarrow \{x_2, y_2(x_1)\} \rightsquigarrow \{x_1, y_1(x_2)\}$ on the level of components. Indeed, choosing the universal quantifiers as $y_1(x_2) = \neg x_2, y_2(x_1) = x_1$ refutes the formula.

▶ **Example 13.** Another example is the false formula $\forall u \exists x (u(x) \leftrightarrow x)$. Informally, it states that every Boolean function has a fixed point. Indeed, u depends on x because x is an argument of u, and so the only component $\{u(x), x\}$ in this formula already forms a $\rightsquigarrow$-loop. (Also, x depends on u as it is quantified after u, but this fact is not required here. The formula $\exists x \forall u (u(x) \leftrightarrow x)$ is false as well.)

We carry the classical approach to the second-order setting, in particular to the fragment of formulae introduced next.

▶ **Definition 14** (Braided formulae)**.** *Let φ be a closed prenex formula, i.e., it is of the form $Q_1 \vec{f_1} \cdots Q_m \vec{f_m}\, \theta$, for θ quantifier-free. Then φ is* braided *if, for every quantifier Q_i, the arguments of each $g \in \vec{f_i}$ are quantified after g*

a) *in the quantifier blocks Q_i and Q_{i+1}, if Q_i is existential, and*

b) *in the quantifier blocks Q_i, Q_{i+1}, and Q_{i+2}, if Q_i is universal.*

Here, we restrict ourselves to braided $\Sigma_\omega^{\mathsf{usk}}$-formulae. That is, we consider only formulae of the form $Q_1 f_1 \cdots Q_m f_m \bigwedge_{i=1}^{k} C_k$, where $C_k = (\ell_k^1 \vee \ell_k^2)$ for literals ℓ_k^1, ℓ_k^2, and where terms do not contain nested proper functions.

Next, we prove that the conditions (1)–(4) are necessary for φ being true in the braided case. Afterwards, we show that they are also sufficient.

▶ **Lemma 15.** *Assume $\varphi \in \Sigma_\omega^{\mathsf{usk}}$ and braided. If any of* (1) *to* (4) *is violated, then φ is false.*

Proof. Let $G = (V, E)$ be the implication graph of φ.

(1) Let u and u' be distinct universal vertices such that (u, u') belongs to the transitive closure of E. Using an interpretation that maps u and u' to the constant functions 1 and 0, respectively, we can conclude that φ cannot be true.

(2) If v and $\neg v$ are vertices from the same component, it follows that φ can be true only if $v \leftrightarrow \neg v$ holds for some interpretation, which is clearly impossible.

(3) Let v and u be an existential and universal vertex from the same component, respectively, such that $v \not\rightsquigarrow u$. Hence u is not an argument of v. We proceed to a case distinction:

- **i)** The function v is quantified before u in φ: By the braided property, all the arguments of v (if there are any) are in the same quantifier block as v, or in the next one. Since changing the ordering of quantifiers in a universally quantified block does not have semantical consequences, we may stipulate that u is the final quantifier of its block. Hence all arguments of v are quantified before u as well. As a consequence, there is a fixed interpretation of terms such that v fully evaluates to either zero or one, but still must equal the universal u which is quantified later, which is impossible.
- **ii)** The function u is quantified before v: Since $v \not\rightsquigarrow u$, there must exist an argument z of u that is not an argument of v and that is quantified in some block strictly after the block where v is quantified (since v is existential). By the braided property, if u is quantified in a block Q_i it follows that v and z are quantified in the blocks Q_{i+1} and Q_{i+2} respectively. Hence z is universal. Similarly to i), the braided property also implies that all arguments of v are quantified in the quantifier blocks Q_{i+1} and Q_{i+2}. Hence using the same argument as in i), we may assume that z is the final quantifier in its block. Now by selecting u to be the projection function for the universally quantified z, we obtain an analogous contradiction as in i).

(4) Suppose there are components $S_1, \ldots, S_n$ such that $S_i \rightsquigarrow S_{i+1}$ for $i \in [n-1]$ and $S_n \rightsquigarrow S_1$. Let each S_i contain a universal vertex u_i and a vertex v_i such that $u_i \rightsquigarrow v_{i+1}$ for $i \in [n-1]$, and $u_n \rightsquigarrow v_1$. We describe choices of the universal quantifiers such that the formula becomes false. For $1 \leq i < n$, we can pick u_i such that it equals v_{i+1}; either as a projection function if v_{i+1} occurs among its arguments, or as a restriction of v_{i+1} to the set of common arguments of u_i and v_{i+1}. In the second case, every argument of v_{i+1} is also one of u_i or is quantified before u_i. Now the components $S_1, \ldots, S_n$ all have to receive the same truth value, regardless of the existential choices. Finally, u_n is picked as the *negation* of v_1, which renders the formula false. ◀

Next we proceed with the converse direction. We assume that the four above conditions are true, and from this construct a satisfying interpretation.

▶ **Lemma 16.** *Assume $\varphi \in \Sigma_\omega^{\mathsf{usk}}$. If* (1)–(4) *are satisfied, then φ is true.*

Proof. For this direction, we can roughly follow Aspvall et al. [2], but have to take into account that the vertices can also be proper functions.

Let $G = (V, E)$ be the implication graph of φ. The idea is to label the graph with truth values. Each component S in the graph is either unmarked, or marked with `true`, `false`, or `contingent`. Marking a component `true` or `false` means that it can in fact receive the corresponding truth value as a constant function, and `contingent` means that its truth depends on other vertices. Universal components are always contingent.

For every component S, the set $\neg S := \{\neg v \mid v \in S\}$ is again a component. Due to (2), S and $\neg S$ are always distinct. Moreover, the implication graph is *skew-symmetric* in the sense that there is an automorphism (modulo flipping all edges) mapping any literal to its negation. The reason is that the implication $\ell \to \ell'$ is clearly equivalent to $\neg\ell' \to \neg\ell$.

We are now in the position to construct an assignment. This assignment will be consistent in the sense that S is marked `true` iff $\neg S$ is marked `false`, and such that it satisfies all clauses due to the property that no path leads from a `true` component marked to a `false` one. First, we mark all universal components as `contingent`. We then consider the existential components in a reverse topological ordering with respect to E (there exists one, for the strongly connected components always induce an acyclic graph). The algorithm marks each component S in this order as follows.

i) If S is already marked, proceed with the next component.
ii) Otherwise S is existential and unmarked, but everything reachable by S is already marked. If S reaches any `contingent` or `false` component, mark it `false`; otherwise mark it `true`.
iii) Mark $\neg S$ the opposite of S.

Now, whenever a component S is `false`, then either (in ii) it reaches some component marked `contingent` or `false`, or (in iii), by skew-symmetry, all components reaching it are `false`. Likewise, if S is `true`, then either (in ii) it reaches only components marked `true`, or (in iii), by skew-symmetry, it can be reached by a `contingent` or `true` component. Also, by condition (1), there is no path from one `contingent` component to another. It can be shown by induction on the steps of the algorithm, that there is no path from a `true` to a `contingent` or `false` component, and also none from a `contingent` to a `false` component.

All components marked `true` or `false` consist of existential vertices, so these can be assigned the corresponding truth assignment. Let us stress that here it suffices to assign constant functions regardless of the actual dependencies of the variables.

Next, fix some interpretation of the universally quantified variables. We continue the algorithm and refine the labeling of the universal components. By (4), it holds that there is no $\rightsquigarrow$-cycle between the components. This implies that there is again a reverse topological ordering $S_1, S_2, \ldots$ of all components, but now in the sense that $S_j \rightsquigarrow S_i$ implies $i < j$. We process all components in this order as follows.

i') If S is not universal, or if it is already marked, proceed with the next component.
ii') Otherwise, let u be the universal vertex in S (which is unique by (1)).
iii') All dependencies of u are already marked `true` or `false`; in particular, all arguments of u have a marked truth value. Change S to `true` if u evaluates to 1 under the corresponding assignment, and otherwise to `false`.
iv') Mark $\neg S$ the opposite of S.

It remains to establish that the interpretations of the existential variables in universal components can be always selected to mimic the truth value of the universal variable of its component. Recall that any existential vertex v in the component S must depend on u due to (3). This means that either (a) v is a function with u as an argument, or (b) v is quantified after u and has as arguments all arguments of u that are quantified in quantifier blocks after v. If (a) is the case, the we interpret v as the projection function for u. If (b) is the case, then there may be some arguments of u which are not arguments of v, but somewhere in the same quantifier block as v. But note that we may stipulate any fixed order of quantification inside a given quantifier block. Here, we assume that, inside a block, variables are quantified such that, for $i < j$, functions in S_i are quantified before functions in S_j. Then any variable that is quantified in the same block as v and is an argument of u but not of v is quantified before v, and hence has a fixed truth value when we give v its interpretation. Let A be the set of common arguments of v and u, and let $\vec{x}$ and $\vec{b}$ be the sequence of the arguments of u that are not in A and the truth values fixed for those vertices before v is interpreted, respectively. Now interpret v as the restriction of u to A with the determined arguments fixed $\vec{x} \mapsto \vec{b}$. In either case, we assigned v such that it equals u.

Since the above cannot introduce any new paths from a `true` component to a `false` component, all clauses of φ are satisfied. ◀

▶ **Theorem 17.** *The truth problem of braided $\Sigma^{\mathsf{usk}}_{\omega}$-formulae is in* **NL***.*

Proof. By the above two lemmas, it suffices to check conditions (1)–(4). But these are simple reachability tests, which are easily solved in non-deterministic logspace. ◀

Next we apply the result to the lowest levels of the second-order quantifier hierarchy, namely Π^{s}_2-formulae and lower. Here, formulae are of the form

$$\forall f_1 \cdots \forall f_n \exists g_1 \cdots \exists g_m \forall x_1 \cdots \forall x_k \, \theta,$$

so the only terms violating the braided property could be of the form $v_1(\ldots, v_2, \ldots)$, where v_2 is quantified before v_1. But then the argument v_2 can be elided from v_1 by Proposition 10. Only for fragments Σ^{s}_2 or higher we can have formulae like $\exists f \forall g \exists x\, f(x)$ which are genuinely not braided, and which cannot be transformed by term elision. Finally, if the propositional quantifier block is existential (in the Π^{s}_1 fragment), we can omit the simpleness constraint due to Corollary 6. This yields the following collection of results, since **NL**-hardness holds already for the satisfiability of classical propositional core formulae (see, e.g., [11, Thm 16.3]).

▶ **Corollary 18.** *Truth of formulae in Σ^{usk}_1, Π^{uk}_1, Π^{usk}_1 or Π^{usk}_2, respectively, is* **NL***-complete. Also, the lower bound still holds for the respective restrictions to core formulae.*

Note that the above proof hinges on the fact that Lemma 15 works only for braided formulae. If we drop this assumption, then the complexity of the truth problem becomes as hard as for arbitrary formulae, as shown in Section 5.

4 Further Upper Bounds

In the previous section, we showed that the first level of the $\mathsf{SO}^{\mathsf{us}}_2$ hierarchy becomes tractable when restricted to Krom formulae. The same holds when restricted to Horn formulae [4]. Next, we consider the question whether these results can be generalized to higher levels of the SO_2 hierarchy. Indeed, we find several cases where the complexity collapses to a lower class. It is worthy to note that such a collapse occurs only if the final propositional quantifier block of a formula is universal, which also is the case, e.g., for the DQBF fragment (cf. Theorem 9). If the final quantifier block is existential, we show later in the next section that no such collapse occurs.

▶ **Theorem 19.** *Let $k > 0$ be even. Then the truth problem of $\Pi^{\mathsf{sk}}_k \cup \Pi^{\mathsf{sh}}_k$ is in Π^{E}_{k-1} and the truth problem of $\Sigma^{\mathsf{sk}}_{k+1} \cup \Sigma^{\mathsf{sh}}_{k+1}$ is in Σ^{E}_k.*

Proof. The following algorithm decides whether a given formula φ is true, if φ is simple and additionally Krom or Horn. Suppose $\varphi \in \Pi_k$ (resp. $\varphi \in \Sigma_{k+1}$).

First we non-deterministically guess in exponential time a truth table for each quantified function, except for the final block of existentially quantified functions, performing $k-2$ (resp. $k-1$) alternations in this process. All so evaluated quantifiers are deleted, and in either case we arrive at a formula φ' of the form $\exists f_1 \cdots \exists f_n \forall x_1 \cdots \forall x_m \theta$ for quantifier-free θ, and some interpretation I for the free variables in φ'. It remains to give a procedure that decides whether $I \vDash \varphi'$. If this part of the algorithm runs in deterministic exponential time w. r. t. $|\varphi|$, then this proves an overall Π^{E}_{k-1} or Σ^{E}_{k} bound, respectively.

To do so, we first perform some simplifications. W.l.o.g. $f_{o+1}, \ldots, f_n$ are propositions and $f_1, \ldots, f_o$ are proper functions, for some $o \in [n]$. We deterministically loop over all possible values for $f_{o+1}, \ldots, f_n$, substitute these in the formula, and remove the quantifiers. This leads only to an exponential factor in the runtime and ensures that all existentially quantified variables are proper functions. By this, we arrive at a Horn or Krom formula

$$\varphi'' = \exists f_1 \cdots \exists f_o \forall x_1 \cdots \forall x_m \theta'$$

for quantifier-free θ'. Note that φ'' may still contain free proper functions. But due to the simpleness condition, and since the f_i are functions as well, no existential variable is nested inside another function. This is crucial for the next step.

We use the *universal expansion* technique, which has been applied to DQBF as well [4]. The idea is to translate the universal quantifiers into an equivalent large conjunction. Let $r_i := \mathrm{ar}(f_i)$. We replace each existential variable f_i by exponentially many propositions $y_{i,\vec{a}}$, one for each possible input tuple $\vec{a} \in \{0,1\}^{r_i}$. For all possible assignments $\vec{b} \in \{0,1\}^m$ to the x_i, we create a modified copy $\theta'[\vec{b}]$ of the matrix θ' defined as follows. If $\vec{b} = (b_1, \ldots, b_m)$, then each x_i is replaced by b_i. Next, all terms t in $\theta'[\vec{b}]$ not containing any f_i are replaced by their valuation $[\![t]\!]_I \in \{0,1\}$. Now all terms are either constant, or have the head f_i and only constant arguments. Finally, the latter terms $f_i(b_1, \ldots, b_{r_i})$ are replaced by the proposition $y_{i,(b_1,\ldots,b_{r_i})}$. The resulting formula is the following:

$$\psi := \underset{\substack{i \in [o] \\ \vec{a} \in \{0,1\}^{r_i}}}{\exists} y_{i,\vec{a}} \bigwedge_{\vec{b} \in \{0,1\}^m} \theta'[\vec{b}]$$

This formula contains no free variables and is true if and only if $I \vDash \varphi''$. In other words, it is a simple propositional formula with existential proposition quantifiers, and its matrix $\bigwedge_{\vec{b} \in \{0,1\}^m} \theta'[\vec{b}]$ is Krom or Horn. Hence the truth of ψ can be computed in deterministic polynomial time w. r. t. $|\psi|$, and consequently in deterministic exponential time w. r. t. $|\varphi|$. ◀

If the non-deterministic part of the algorithm, the guessing of all quantified functions but the last block, is removed, then we obtain a deterministic exponential time algorithm for Σ_1^{sh}-formulae. the

▶ **Theorem 20.** *Truth of* Σ_1^{sh} *is in* **EXP**.

In fact, we can combine this approach with the **NL** algorithm from Section 3 as well:

▶ **Theorem 21.** *Truth of* Σ_1^{sk} *is in* **PSPACE**.

Proof. Given a formula $\varphi \in \Sigma_1^{\mathsf{sk}}$, we run the reachability algorithm from Section 3 on the formula ψ that would result from the translation in Theorem 19. However, instead of expanding φ to ψ first, which would require exponential space, we perform the reachability tests in polynomial space, constructing only the needed parts of ψ on-the-fly. ◀

Observe why the technique relies on the final quantifier block being universal: otherwise the resulting formula $\bigvee_{\vec{b} \in \{0,1\}^m} \theta'[\vec{b}]$ would not be in CNF, and hence neither Horn nor Krom.

5 Lower bounds

In the previous sections, we showed that the complexity of a fragment sometimes decreases when restricted to Horn or Krom matrix, when compared to the general fragment with the same quantifier prefix. However, in many cases the complexity stays the same. Often the logics are powerful enough to simulate specific Boolean connectives, such as disjunction and negation, in terms of quantified Boolean functions. In these cases, the whole Boolean part of the formula can essentially be reduced to unit clauses, which of course renders the Horn and Krom restriction meaningless.

5.1 Cases with an existential function quantifier

The first result of this section is also the most general; it concerns all non-simple formulae for quantifier prefixes that include Σ_1 – that is, everything but Π_1. (Recall that simple and non-simple Π_1 are equivalent.) By the introduction of additional existential functions that simulate disjunction and negation, we bring an arbitrary CNF into core form. This is stated in the following lemma, of which the proof can be found in Appendix C.

▶ **Lemma 22.** *Let $\mathcal{Q}\,\theta$ be a formula in CNF, with θ quantifier-free in CNF and $\mathcal{Q}$ being a sequence of quantifiers. Then $\mathcal{Q}\,\theta$ is equivalent to a logspace-computable formula $\exists\vec{f}\,\mathcal{Q}\,\forall\vec{y}\,\exists\vec{z}\theta'$ in CNF such that θ' is quantifier-free, $\vec{f}$ are function symbols, and $\vec{y}$, $\vec{z}$ are propositions. Moreover, if θ has uniqueness, then so has θ'.*

▶ **Theorem 23.** *For $k \geq 1$, truth of Σ_k^{uc} is Σ_k^{E}-complete.*

Proof. The upper bound is due to Theorem 7. For the lower bound, we use Lemma 22 and reduce from Σ_k, for which the truth is Σ_k^{E}-complete by Theorem 7. Let

$$\varphi = \exists\vec{f}_1\forall\vec{f}_2\cdots Q_k\vec{f}_k\,Q_{k+1}\vec{x}\,\theta$$

be given, where θ is quantifier-free, each $\vec{f}_i$ is a sequence of functions, and $\vec{x}$ is a sequence of propositions.

The first step is to transform φ to an equivalent formula with uniqueness. For any function h that violates uniqueness, we introduce fresh distinct copies $h_1, \ldots, h_n$ of h, for each distinct tuple of arguments $\vec{a}_1 \ldots \vec{a}_n$ of h occurring in φ, together with distinct fresh propositional variables $\vec{z}, \vec{z}_1, \ldots, \vec{z}_n$. We then append subformulae to φ whose purpose is to state that the interpretations of h_i and h coincide. If $Q_k = \exists$ and $Q_{k+1} = \forall$, we modify φ such that $\forall\vec{x}\,\theta$ is replaced with

$$\exists h_1 \ldots h_n \forall\vec{x}\,\vec{z}\,\vec{z}_1 \ldots \vec{z}_n\Big(\bigwedge_{i\in[n]}\big((\vec{z} \leftrightarrow \vec{z}_i) \to (h(\vec{z}) \leftrightarrow h_i(\vec{z}_i))\big)\Big) \land \Big(\big(\bigwedge_{i\in[n]}(\vec{z}_i \leftrightarrow \vec{a}_i)\big) \to \theta^*\Big),$$

where θ^* is obtained from θ by replacing the occurrences of $h(\vec{a}_i)$ by $h_i(\vec{z}_i)$, for each $i \in [n]$. On the other hand, if $Q_k = \forall$ and $Q_{k+1} = \exists$, we modify φ such that $\exists\vec{x}\,\theta$ is replaced with

$$\forall h_1 \ldots h_n \exists\vec{x}\,\vec{z}\,\vec{z}_1 \ldots \vec{z}_n\Big(\bigvee_{i\in[n]}\big((\vec{z} \leftrightarrow \vec{z}_i) \land (h(\vec{z}) \leftrightarrow \neg h_i(\vec{z}_i))\big)\Big) \lor \Big(\big(\bigwedge_{i\in[n]}(\vec{z}_i \leftrightarrow \vec{a}_i)\big) \land \theta^*\Big),$$

where θ^* is as above.

The second step is to establish CNF. It is folklore that arbitrary formulae can be translated into an equivalent CNF with the introduction of additional existentially quantified propositions after the final quantifier block $\vec{x}$. If k is odd, these existential propositions can be pulled in front of $\vec{x}$ (by increasing their arity and adding $\vec{x}$ as their parameter) and added to the (existential) block $\vec{f_k}$. If k is even this step can be skipped since $\vec{x}$ is existential as well. Hence we can assume that θ is in CNF and has uniqueness.

It remains to conduct the final translation into core clauses. For any Σ_k^{u}-formula $\varphi' = \exists \vec{f_1} \forall \vec{f_2} \cdots Q_k \vec{f_k} Q_{k+1} \vec{x}\, \theta'$, we can apply Lemma 22 to the subformula after $\exists \vec{f_1}$ and obtain an equivalent formula $\exists \vec{f_1} \exists \vec{g} \forall \vec{f_2} \cdots Q_k \vec{f_k} Q_{k+1} \vec{x} \forall \vec{y} \exists \vec{z}\, \theta''$ where θ'' is a quantifier-free CNF with uniqueness. Using the same argument as above, if $Q_{k+1} = \forall$, then we can eliminate the first-order alternation by transforming the $\vec{z}$ into existentially quantified functions depending on both $\vec{x}$ and $\vec{y}$ and adding them to the block $\vec{f_k}$. Otherwise, if $Q_{k+1} = \exists$, then we instead transform the $\vec{y}$ into functions depending on $\vec{x}$ and move them into the universal block $\vec{f_k}$. In each case, we arrive at an Σ_k^{uc}-formula. Note we do not consider the simpleness property at this point, since this step may produce new function symbols that appear nested in other functions. ◀

▶ **Theorem 24.** *For $k \geq 2$, truth of Π_k^{uc} is Π_k^{E}-complete.*

Proof. The proof is the same as for Theorem 23, except that in the last step, the formula is of the form

$$\forall \vec{f_1} \exists \vec{f_2} \forall \vec{f_3} \cdots Q_k \vec{f_k} Q_{k+1} \vec{x}\, \theta'$$

and we apply the lemma to the subformula after $\exists \vec{f_2}$. ◀

Lemma 22, used in the above reductions, introduces existential quantifiers that are not braided. Compared to the previous section, this small difference leads from **NL**-membership to Σ_k^{E}-completeness. If the final proposition block is existential *and* there is at least one existential function block, the result carries over even with simpleness due to Corollary 6:

▶ **Theorem 25.** **1.** *Let $k > 0$ be even. The truth problem of Σ_k^{usc} is Σ_k^{E}-complete and the truth problem of Π_{k+1}^{usc} is Π_{k+1}^{E}-complete.*
2. *The truth problem of $\Sigma_\omega^{\mathsf{usc}}$ is* **AEXP**(poly)*-complete.*

What if the proposition block is universal, i.e., k is odd for Σ_k and even for Π_k? Then, as shown in Theorem 19, we fall down one level in the hierarchy. Hardness results follow from the observation that Σ_k (Π_k, resp.) is a syntactic fragment of Σ_{k+1} (Π_{k+1}, resp.).

▶ **Theorem 26.** *Let $k > 2$ be odd. The truth problem of Σ_k^{usc} is $\Sigma_{k-1}^{\mathrm{E}}$-complete and the truth problem of Π_{k+1}^{usc} is Π_k^{E}-complete.*

5.2 The fragment Π_1 without uniqueness

We established the **NL** upper bound of Π_1 if we have uniqueness and Krom (Corollary 18); the case with uniqueness and Horn is open. Here, we proceed with Π_1 without uniqueness. As we have no existential function quantifiers, the reduction from before does not apply. Nonetheless, it turns out that this fragment is still as hard as the full logic Π_1.

▶ **Theorem 27.** *Truth of Π_1^{sc}-formulae is Π_1^{E}-hard.*

Proof. We reduce from the truth of arbitrary Π_1-formulae, which by Theorem 7 is Π_1^{E}-complete. Hence let φ be a Π_1-formula, i.e.,

$$\varphi = \forall f_1 \cdots \forall f_n \exists x_1 \cdots \exists x_m\, \theta$$

for function variables $f_1, \ldots, f_n$, propositions $x_1, \ldots, x_m$, and θ quantifier-free. Since the propositional quantifier block is existential, we can w.l.o.g. assume that θ is in 3CNF.[1]

The idea is to add $\forall g$ to the beginning of the formula, where g is a fresh binary function symbol, and to express in the reduction that g is the *nand* function, i.e., $g(b_1, b_2) = 1 - b_1 b_2$. In what follows, we use the constants 0 and 1, which can easily be simulated by adding new propositional quantifiers $\exists z_0 \exists z_1$ and unit clauses $\neg z_0 \land z_1$. To describe the behaviour of g, we add existentially quantified propositions d, d', e, e' and the following core clauses:

$$D_1 := g(0,0) \to d, \quad D_2 := g(0,0) \to e, \quad D_3 := g(d,0) \to d', \quad D_4 := g(0,e) \to e'.$$

Furthermore, every clause $C := (\ell_1 \lor \ell_2 \lor \ell_3)$ of θ is replaced by $e' \to g(d', C^*)$, where C^* is a *nand*-expression equivalent to $\neg C$, using g as a symbol for *nand*.[2] C^* has length $\mathcal{O}(|C|)$.

Call the resulting formula θ^*. To prove the correctness of the reduction, we show that θ is equivalent to $\theta' := \forall g \exists d \, \exists d' \, \exists e \, \exists e' (\bigwedge_{i=1}^{4} D_i \land \theta^*)$.

The easy direction is from right to left: Since g is universal, in particular we can assume that g is *nand*. As $g(0,0) = g(1,0) = g(0,1) = 1$, the propositions d, e, d', e' must all be true. Since also all clauses of the form $e' \to g(d', C^*)$ are true by assumption, C^* is false. Consequently, C is true.

For the converse direction, let g be arbitrary. We define suitable witnesses for d, e, d' and e'.

- If $g(0,0) = 0$, then we set $d, e, d', e' := 0$, which satisfies all clauses of the form $e' \to g(d', C^*)$, as well as $D_1, \ldots, D_4$.
- If $g(0,1) = 0$, then we can similarly set $d, d', e := 1$ and $e' := 0$.
- Otherwise $g(0,0) = g(0,1) = 1$. Here, we must set $e, e', d := 1$.
  - If $g(1,0) = 0$, then we set $d' := 0$. Then $g(d', C^*) = g(0, C^*) = 1$ regardless of C^*.
  - If $g(1,0) = 1$, then we set $d' := 1$.
    - If $g(1,1) = 1$, then g is constant one, and the terms $g(d', C^*)$ are trivially true.
    - If $g(1,1) = 0$, then g is the actual *nand* function, and $g(d', C^*) \equiv \neg(1 \land C^*) \equiv C$ is true by assumption.

Finally, we replace θ by θ' in φ, move $\forall g$ to the front of the formula, and obtain simpleness of the formula by Corollary 6. ◀

The above results easily "relativize" to the case of more quantifier alternations before the final universal function quantifier block:

▶ **Theorem 28.** *Let $k > 0$ be odd. Then the truth of $\Sigma_{k+1}^{\mathsf{sc}}$ is $\Sigma_{k+1}^{\mathrm{E}}$-complete, and the truth of Π_k^{sc} is Π_k^{E}-complete.*

5.3 The Σ_1 cases with simpleness but no uniqueness

Curiously, while Π_1^{sc} is Π_1^{E}-complete, its dual fragment Σ_1^{sc} is likely easier than Σ_1^{E}, although harder than Σ_1^{usc}. We consider these final fragments in this subsection.

▶ **Theorem 29.** *Truth of formulae in Σ_1^{sc} or Σ_1^{sk} is* **PSPACE***-complete.*

Proof. The upper bound is given by Theorem 21. We show the hardness for Σ_1^{sc}, which implies the lower bound for Σ_1^{sk}. Let M be a single-tape Turing machine that decides some **PSPACE**-complete problem in deterministic space $p(n)$, where $p(n) \geq n$ is some polynomial.

[1] The approach is the same as for the classical reduction from SAT to 3SAT and can be found in standard textbooks (e.g. [1, Lemma 2.14]).

[2] We can choose for example $C^* := h(g(h(g(h(\ell_1), h(\ell_2))), h(\ell_3)))$, where $h(\varphi) = g(\varphi, \varphi)$.

W.l.o.g., we may assume that the computation of M halts in time $g(n)$ by reaching a unique rejecting or a unique accepting configuration, where $g(n)$ is some exponential function. For each input x, we compute a formula φ in logspace that is true iff M accepts x. The formula φ will be of the form

$$\exists f\, \forall v_1 \cdots \forall v_m\, \theta,$$

where θ is quantifier-free, simple and core, f is a function variable, and the v_i are propositions. Thus $\varphi \in \Sigma_1^{\mathsf{sc}}$.

If M has states Q and tape alphabet Γ, then a configuration of M is a triple (h, q, w), where $h \in [p(n)]$ denotes the head position on the tape, $q \in Q$ is the state of the machine, and $w \in \Gamma^{p(n)}$ is the tape content. We stipulate an arbitrary coding function $\langle\cdot\rangle : Q \cup \Gamma \to \{0,1\}^k$ that expands each state and each tape symbol to a fixed-width binary vector. For tape positions $j \in [p(n)]$, we use the unary encoding $\mathsf{bit}(j) := (0^{j-1}10^{p(n)-j})$. Using the coding function $\langle\cdot\rangle$, configurations of M can be now presented as binary strings of length $p(n) + k + kp(n)$.

The idea behind φ is as follows: The function f is used to encode a set of (binary encodings of) configurations of M. In order to take a head position, a state, and a tape content as an argument, the function f will have arity $p(n) + k + kp(n)$. In θ, we stipulate that f contains the initial configuration and is closed under transitions of M, but does not reach the unique rejecting configuration. Hence it expresses that M accepts x, as desired.

We will next describe θ more formally. Let M have initial state $q_0 \in Q$, and let $x = x_1 \cdots x_n$. First, we define the formula ψ_1 expressing that f contains the initial configuration:

$$\psi_1 := f(\mathsf{bit}(1); \langle q_0\rangle ; \langle x_1\rangle \cdots \langle x_n\rangle\ \langle \square\rangle \cdots \langle \square\rangle\,),$$

where $\square \in \Gamma$ denotes the special symbol for blank. Next, ψ_2 states that f is closed under transitions of M (f may contain superfluous configurations, but this does not hurt the correctness of the reduction). Let $\delta \colon Q \times \Gamma \to Q \times \Gamma \times \{-1, 0, 1\}$ be the transition function of M; e.g., if $\delta(q, a) = (q', b, -1)$, then M upon reading a in state q writes b, enters state q', and moves the head to the left. Define

$$\psi_2 := \forall \vec{v} \bigwedge_{\substack{j \in [p(n)] \\ \delta(q,a) = (q',a',i) \\ 1 \le j+i \le p(n)}} \Big(f(\mathsf{bit}(j); \langle q\rangle ; v_1 \cdots v_{k(j-1)}\ \langle a\rangle\ v_{kj+1} \cdots v_{kp(n)}) \\ \to f(\mathsf{bit}(j+i); \langle q'\rangle ; v_1 \cdots v_{k(j-1)}\ \langle a'\rangle\ v_{kj+1} \cdots v_{kp(n)})\Big),$$

where $\forall \vec{v}$ denotes $\forall v_1 \cdots \forall v_{kp(n)}$.

Finally, it remains to express that the rejecting configuration cannot be reached, which w.l.o.g. is a blank tape with M's head on the first position and in a designated state $q_r \in Q$.

$$\psi_3 := \neg f(\mathsf{bit}(1); \langle q_r\rangle ; \langle \square\rangle \cdots \langle \square\rangle\,)$$

By pulling the quantifiers in ψ_2 to the front, it is straightforward to see that $\exists f(\psi_1 \wedge \psi_2 \wedge \psi_3)$ is equivalent to a Σ_1-formula with only core clauses and with no nesting of functions, i.e., to a Σ_1^{sc}-formula. ◀

The proof of the following theorem is similar to that of Theorem 29. However, as an exponential time computation may require exponential space, some more care is required for the encodings. The computation is now encoded with a function that takes a tape address and the current timestep as arguments rather than the whole tape content. A detailed proof of the theorem can be found in Appendix D.

▶ **Theorem 30.** *Truth of formulae in* Σ_1^{sh} *is* **EXP***-complete.*

6 Summary

In this article, we studied the second-order quantifier hierarchy of Boolean logic. Boolean second-order logic, where quantifiers range over Boolean functions instead of mere propositions, can be seen as a generalization of logics such as DQBF that offer fine-grained control of dependencies between variables. Here, we turned to certain fragments where the propositional part is restricted to either Horn, Krom, or core formulae. Moreover, we introduced and considered two natural restrictions of second-order term constructions, namely simpleness (where proper function symbols cannot occur nested) and uniqueness (where all occurrences of a function have the same arguments). Using this terminology, DQBF is simple unique Σ_1.

We considered all possible combinations of these restrictions with respect to each level of the quantifier hierarchy, and obtained an almost complete classification of the computational complexity of the respective decision problem (cf. Table 1 on page 3). In almost all cases we obtained completeness results (with respect to logspace reductions). We showed that the complexity of Σ_1 and Π_1 formulae in Horn and/or Krom form collapse down to one of several classes that range from **NL** over **PSPACE** to **EXP**. Curiously, core Σ_1 stays Σ_1^{E}-hard if we lack simpleness, while core Π_1 stays Π_1^{E}-hard if we lack uniqueness. Moreover, Π_2 stays in **NL** if simple, unique, and Krom. For $k \geq 3$, for all considered restrictions to Σ_k (Π_k, resp.) the complexity either stays Σ_k^{E}-complete (Π_k^{E}-complete, resp.) or drops one level down to $\Sigma_{k-1}^{\mathrm{E}}$ (Π_{k-1}^{E}, resp.) depending on uniqueness, simpleness, and whether k is even or odd. Furthermore, a direct corollary of the aforementioned results is that the complexity of $\Sigma_\omega^{\mathsf{usc}}$-formulae is **AEXP**(poly)-complete.

For the upper bounds, we mostly utilized generalizations of existing **NL** or **P** algorithms for classical Krom or Horn formulae. For the lower bounds, we introduced a number of different techniques; the common scheme being that one can exploit the ability to quantify functions to nullify the Horn and/or Krom restriction.

The most notable open case is that of simple unique Horn Π_1, which we conjecture to be **P**-complete, dually to the **P**-complete Σ_1 case (that is, DQBF-Horn [4]). Moreover, by Corollary 6, *non-simple* unique Π_1 has the same complexity. The final missing case, simple unique Π_2, likely reduces to these basic cases, but its complexity stays an open question for now as well.

References

1 Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. URL: `http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264`.

2 Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inf. Process. Lett.*, 8(3):121–123, 1979. `doi:10.1016/0020-0190(79)90002-4`.

3 Herbert Baier and Klaus W. Wagner. The Analytic Polynomial-Time Hierarchy. *Mathematical Logic Quarterly*, 44(4):529–544, 1998. URL: `http://onlinelibrary.wiley.com/doi/10.1002/malq.19980440412/abstract`.

4 Uwe Bubeck and Hans Kleine Büning. Dependency quantified horn formulas: Models and complexity. In *SAT*, volume 4121 of *Lecture Notes in Computer Science*, pages 198–211. Springer, 2006.

5 Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981. `doi:10.1145/322234.322243`.

6 Miika Hannula, Juha Kontinen, Martin Lück, and Jonni Virtema. On quantified propositional logics and the exponential time hierarchy. In Domenico Cantone and Giorgio Delzanno, editors,

Proceedings of the Seventh International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2016, Catania, Italy, 14-16 September 2016, volume 226 of *EPTCS*, pages 198–212, 2016. `doi:10.4204/EPTCS.226.14`.

7 Juris Hartmanis, Neil Immerman, and Vivian Sewelson. Sparse sets in NP-P: EXPTIME versus NEXPTIME. *Information and Control*, 65(2/3):158–181, 1985.

8 Markus Lohrey. Model-checking hierarchical structures. *J. Comput. Syst. Sci.*, 78(2):461–490, 2012. `doi:10.1016/j.jcss.2011.05.006`.

9 Martin Lück. Complete problems of propositional logic for the exponential hierarchy. *CoRR*, abs/1602.03050, 2016.

10 Pekka Orponen. Complexity classes of alternating machines with oracles. In Josep Díaz, editor, *Automata, Languages and Programming, 10th Colloquium, Barcelona, Spain, July 18-22, 1983, Proceedings*, volume 154 of *Lecture Notes in Computer Science*, pages 573–584. Springer, 1983. `doi:10.1007/BFb0036938`.

11 Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

12 Gary L. Peterson and John H. Reif. Multiple-person alternation. In *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*, pages 348–363. IEEE Computer Society, 1979. `doi:10.1109/SFCS.1979.25`.

13 Gary L. Peterson, John H. Reif, and Salman Azhar. Lower bounds for multiplayer noncooperative games of incomplete information. *Computers & Mathematics with Applications*, 41(7):957–992, 2001. `doi:10.1016/S0898-1221(00)00333-3`.

14 Christoph Scholl and Ralf Wimmer. Dependency quantified boolean formulas: An overview of solution methods and applications - extended abstract. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, volume 10929 of *Lecture Notes in Computer Science*, pages 3–16. Springer, 2018. `doi:10.1007/978-3-319-94144-8_1`.

15 Ankit Shukla, Armin Biere, Luca Pulina, and Martina Seidl. A survey on applications of quantified boolean formulas. In *31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, Portland, OR, USA, November 4-6, 2019*, pages 78–84. IEEE, 2019. `doi:10.1109/ICTAI.2019.00020`.

16 Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976. `doi:10.1016/0304-3975(76)90061-X`.

17 Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time: Preliminary report. In Alfred V. Aho, Allan Borodin, Robert L. Constable, Robert W. Floyd, Michael A. Harrison, Richard M. Karp, and H. Raymond Strong, editors, *Proceedings of the 5th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1973, Austin, Texas, USA*, pages 1–9. ACM, 1973. `doi:10.1145/800125.804029`.

A Complexity toolbox

Alternating machines

We assume the reader to be familiar with basic complexity classes and notions such as Turing machines (TMs). We follow the definition of *alternating* TMs by Chandra et al. [5]. The states Q of such an alternating machine (ATM) are divided into disjoint sets $Q_\exists$ of *existential states* and $Q_\forall$ of *universal states*. Also, Q contains a designated *initial* state q_i, an *accepting* state q_a and a *rejecting* state q_r, where w.l.o.g. the initial state is always existential. A transition from an existential to a universal state, or vice versa, is called *alternation*. In this setting, a *non-deterministic* machine is one that never alternates, and a *deterministic* machine is one that provides at most one valid transition for every configuration.

A configuration is *accepting in k steps* if it contains no rejecting state, and furthermore either it contains an accepting state, or, provided $k > 0$, it contains an existential state and has a valid transition to a configurations that accepts in $k - 1$ steps, or contains an universal state and every valid transition leads to a configuration that accepts in $k-1$ steps. The *language decided by M* is the set of all inputs such that the initial configuration is accepting in k steps for some k.

As usual, the classes $\mathbf{EXP}$ and $\mathbf{NEXP}$ contain those problems which are decidable by a (non-)deterministic machine in time $2^{p(n)}$, for some polynomial p. Given a complexity class $\mathcal{C}$, its complement class is denoted by $\mathbf{co}\mathcal{C}$.

▶ **Definition 31.** *For $g(n) \geq 1$, the class $\mathbf{ATIME}(t(n), g(n))$ consists of the problems A for which there is an ATM deciding A in time $\mathcal{O}(t(n))$ with at most $g(n)-1$ alternations on inputs of length n.*

▶ **Definition 32.** *For function classes $\mathcal{F}, \mathcal{G}$,*

$$\mathbf{ATIME}(\mathcal{F}, \mathcal{G}) := \bigcup_{f \in \mathcal{F}, g \in \mathcal{G}} \mathbf{ATIME}(f(n), g(n)).$$

▶ **Definition 33.**

$$\mathbf{AEXP} := \mathbf{ATIME}(2^{n^{\mathcal{O}(1)}}, 2^{n^{\mathcal{O}(1)}}), \qquad \mathbf{AEXP}(\text{poly}) := \mathbf{ATIME}(2^{n^{\mathcal{O}(1)}}, n^{\mathcal{O}(1)}).$$

Oracle machines

An *oracle Turing machine* is a Turing machine that additionally has an access to an *oracle* set B. The machine can query B by writing an instance x on a designated *oracle tape* and moving to a *query state* $q_?$. In the next configuration one of two states q_+ and q_- is assumed depending on whether $x \in B$ or not. There is no bound on the number of oracle queries during a computation of an oracle machine; the machine can erase the oracle tape and pose more queries.

If B is a language, then the usual complexity classes $\mathbf{P}, \mathbf{NP}, \mathbf{NEXP}$ etc. are generalized to $\mathbf{P}^B, \mathbf{NP}^B, \mathbf{NEXP}^B$ etc. where the definition is just changed from ordinary Turing machines to corresponding oracle machines with an oracle for B. If $\mathcal{C}$ is a class of languages, then $\mathbf{P}^{\mathcal{C}} := \bigcup_{B \in \mathcal{C}} \mathbf{P}^B$ and so on.

▶ **Definition 34** (The Polynomial Hierarchy [16]). *The levels of the polynomial hierarchy are defined inductively, where $k \geq 1$:*

- $\Sigma_0^{\mathrm{P}} = \Pi_0^{\mathrm{P}} = \Delta_0^{\mathrm{P}} := \mathbf{P}$.
- $\Sigma_k^{\mathrm{P}} := \mathbf{NP}^{\Sigma_{k-1}^{\mathrm{P}}}$, $\Pi_k^{\mathrm{P}} := \mathbf{coNP}^{\Sigma_{k-1}^{\mathrm{P}}}$, $\Delta_k^{\mathrm{P}} := \mathbf{P}^{\Sigma_{k-1}^{\mathrm{P}}}$.

▶ **Definition 35** (The Exponential Hierarchy [7]). *The levels of the exponential hierarchy are defined inductively, where $k \geq 1$:*

- $\Sigma_0^{\mathrm{E}} = \Pi_0^{\mathrm{E}} = \Delta_0^{\mathrm{E}} = \mathbf{EXP}$.
- $\Sigma_k^{\mathrm{E}} := \mathbf{NEXP}^{\Sigma_{k-1}^{\mathrm{P}}}$, $\Pi_k^{\mathrm{E}} := \mathbf{coNEXP}^{\Sigma_{k-1}^{\mathrm{P}}}$, $\Delta_k^{\mathrm{E}} := \mathbf{EXP}^{\Sigma_{k-1}^{\mathrm{P}}}$.

▶ **Theorem 36** ([5]). *For all $k \geq 1$:*

$$\Sigma_k^{\mathrm{P}} = \mathbf{ATIME}(n^{\mathcal{O}(1)}, k), \qquad \Pi_k^{\mathrm{P}} = \mathbf{co}\Sigma_k^{\mathrm{P}}.$$

Just as for the polynomial hierarchy, two competing definitions of Σ_k^{E} exist in the literature, one in terms of oracles and one as the class $\mathbf{ATIME}(2^{n^{\mathcal{O}(1)}}, k)$ [3, 8, 10].

▶ **Theorem 37** ([10]). *For all $k \geq 1$:*

$$\Sigma_k^{\mathrm{E}} = \mathbf{ATIME}(2^{n^{\mathcal{O}(1)}}, k), \qquad \Pi_k^{\mathrm{E}} = \mathbf{coATIME}(2^{n^{\mathcal{O}(1)}}, k).$$

A *logspace-reduction* from A to B is a logspace computable function f such that $x \in A \Leftrightarrow f(x) \in B$. If such f exists then A is *logspace-reducible* to B, in symbols $A \leq_{\mathrm{m}}^{\log} B$. If $A \in \mathcal{C}$ implies $A \leq_{\mathrm{m}}^{\log} B$, then B is $\leq_{\mathrm{m}}^{\log}$*-hard* for $\mathcal{C}$, and B is $\leq_{\mathrm{m}}^{\log}$*-complete* for $\mathcal{C}$ if $B \in \mathcal{C}$ and B is $\leq_{\mathrm{m}}^{\log}$-hard for $\mathcal{C}$. In this paper all reductions are logspace-reductions if not stated otherwise.

B Proof of Proposition 10

▶ **Proposition 10** (Free term elision). *Let $\varphi \in \mathsf{SO}_2^{\mathrm{u}}$ be a prenex formula, f a function variable not free in φ, and t a term free in φ. Then eliding t from f yields a formula equivalent to φ.*

Proof. Assume that φ, f and t are as above, and that $\mathrm{ar}(f) = n$ and g is a variable of arity $n-1$ that does not appear in φ. We prove that eliding the i-th argument of f yields an equivalent formula, where i is any position such that the i-th argument of f is t.

For a function F and $b \in \{0,1\}$, define the $(n-1)$-ary function

$$F_{|b}(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n) := F(a_1, \dots, a_{i-}, b, a_{i+1}, \dots, a_n).$$

Also, let $\varphi^\star$ be the formula φ with the i-th argument of f elided, i.e., f replaced by g and the i-th argument deleted in any occurrence of f as a term. For an interpretation I, define $I^\star$ like I except that $I^\star(g) := I(f)_{|I(t)}$. We show by induction on φ that $I(\varphi) = I^\star(\varphi^\star)$ for all interpretations I. It is easy to see that this proves the claim from the beginning, where neither f nor g appears free.

If φ does not contain t, and hence f, then we are done. Otherwise, if φ is of the form $f(t_1, \dots, t_{i-1}, t, t_{i+1}, \dots, t_n)$, then clearly

$$\begin{aligned} I(\varphi) &= I(f)(I(t_1), \dots, I(t_{i-1}), I(t), I(t_{i+1}), \dots, I(t_n)) \\ &= I(f)_{|I(t)}(I(t_1), \dots, I(t_{i-1}), I(t_{i+1}), \dots, I(t_n)) \\ &= I^\star(g)(I^\star(t_1), \dots, I^\star(t_{i-1}), I^\star(t_{i+1}), \dots, I^\star(t_n)) = I^\star(\varphi^\star). \end{aligned}$$

The inductive steps for applying function variables $h \neq f$, as well as for the Boolean connectives $\wedge$ and $\neg$, are straightforward. Also, the $\forall$-case can be reduced to $\exists$. It remains to consider the $\exists$-case. We divide this into the case where f is quantified and the case where any other function variable $h \neq f$ is quantified.

First, suppose $\varphi = \exists h \psi$, where $h \neq f$. Then whenever $I_H^h \vDash \psi$ for some I and H we have $(I^\star)_H^h = (I_H^h)^\star \vDash \psi^\star$, so $I^\star \vDash \varphi^\star$. Likewise, whenever $I_H^h \vDash \psi^\star$ for some I, then I_H^h is of the form $(J^\star)_H^h = (J_H^h)^\star$ for some J, so $J \vDash \varphi$.

Finally, let $\varphi = \exists f \psi$. If $I_F^f \vDash \psi$ for some I and F, then $(I_F^f)^\star \vDash \psi^\star$ by induction hypothesis. By definition, I^* and $(I_F^f)^\star$ agree everywhere except on f and g, and f does not occur in $\psi^\star$, so $I^* \vDash \exists g \psi^\star = \varphi^\star$ follows.

Suppose that conversely $I^\star \vDash \varphi^\star = \exists g \psi^\star$, so $(I^\star)_G^g \vDash \psi^\star$ for some I and G. As $(I^\star)_G^g = I_G^g$, also $I_G^g \vDash \psi^\star$. Define a function F from G as follows: Let $F(a_1, \dots, a_{i-1}, b, a_{i+1}, \dots, a_n) := G(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n)$ for both $b = 0$ and $b = 1$. Since f does not occur in $\psi^\star$, we can add it to any interpretation, so clearly $(I_F^f)_G^g \vDash \psi^\star$. Now notice that $G = F_{|0} = F_{|1} = F_{|I(t)}$. But then $(I_F^f)_G^g = (I_F^f)^\star$, so by induction hypothesis $I_F^f \vDash \psi$. Hence $I \vDash \exists f \psi = \varphi$. ◀

C Proof of Lemma 22

▶ **Lemma 22.** *Let $\mathcal{Q}\,\theta$ be a formula in CNF, with θ quantifier-free in CNF and $\mathcal{Q}$ being a sequence of quantifiers. Then $\mathcal{Q}\,\theta$ is equivalent to a logspace-computable formula $\exists \vec{f}\, \mathcal{Q} \forall \vec{y}\, \exists \vec{z} \theta'$ in CNF such that θ' is quantifier-free, $\vec{f}$ are function symbols, and $\vec{y}$, $\vec{z}$ are propositions. Moreover, if θ has uniqueness, then so has θ'.*

Proof. We begin with the case where $\mathcal{Q}$ is empty. The other cases are proved analogously.

Accordingly, let θ be of the form $\bigwedge_{i\in[n]} C_i$ with clauses $C_i = \ell_1^i \vee \cdots \vee \ell_r^i$ and the ℓ_j^i being literals (i.e., terms or their negations). The idea of the proof is that all clauses C_i can be reformulated in terms of fresh Boolean functions h_i that act as disjunctions. Thus each clause becomes a single unit (and thus core) clause. Some auxiliary clauses are added in order to properly fix the disjunction as the interpretation of h_i.

We proceed as follows. First, for every literal ℓ in a clause C_i of θ, we introduce a fresh proposition p_ℓ that shall mirror ℓ and can appear inside h_i, since terms can only have other terms as their arguments, and not negations thereof. Next, we introduce a single proposition b the role of which we will explain below. Any clause $C_i = \ell_1^i \vee \cdots \vee \ell_r^i$ is now replaced by the following conjunction $\xi(C_i)$ of core clauses:

$$\xi(C_i) := (b \leftrightarrow h_i(p_{\ell_1^i}, \ldots, p_{\ell_r^i})) \wedge \bigwedge_{k\in[r]} (p_{\ell_k^i} \rightarrow h_i(p_{\ell_1^i}, \ldots, p_{\ell_r^i}))$$

Let us start with the large conjunction on the right hand side: It ensures that the term $h_i(p_{\ell_1^i}, \ldots, p_{\ell_r^i})$ is true if any of its arguments is true. But in order to truly simulate C_i with h_i, we also need to achieve the converse. Otherwise h_i could still be a constant. However, we will quantify b universally, with the effect that the left hand side requires h_i to assume each value, zero and one, for some input. The value $h_i(p_{\ell_1^i}, \ldots, p_{\ell_r^i}) = 0$ can then only be assumed when all $p_{\ell_j^i}$ are zero, as required.

Furthermore, we need to impose some constraints on the proxies p_ℓ, so that p_ℓ in fact mirrors ℓ. In particular, exactly one of p_ℓ and $p_{\neg\ell}$ must be true. For every term t in θ, let g_t be another fresh binary function variable. Define

$$\begin{aligned}\tau(t) := \quad & (b \leftrightarrow g_t(p_t, p_{\neg t})) \wedge (p_t \rightarrow g_t(p_t, p_{\neg t})) \wedge (p_{\neg t} \rightarrow g_t(p_t, p_{\neg t})) \\ & \wedge (\neg p_t \vee \neg p_{\neg t}) \wedge (p_t \rightarrow t) \wedge (p_{\neg t} \rightarrow \neg t).\end{aligned}$$

Here, the first line again ensures that $g_t(p_t, p_{\neg t})$ is true if and only if p_t or $p_{\neg t}$ is true. So, when $b = 1$ then $g_t(p_t, p_{\neg t}) = 1$ and hence we know that at least one of p_t and $p_{\neg t}$ is true. The second line claims that at most one of them is true, and that this reflects the actual value of t. Observe that all used clauses are in core form.

Let now $t_1 \cdots t_s$ be a list of all terms occurring in the clauses of θ. Altogether, we translate $\theta = \bigwedge_{i\in[n]} C_i$ to φ as follows:

$$\varphi := \underset{i\in[n]}{\exists} h_i \underset{i\in[s]}{\exists} g_{t_i} \; \forall b \underset{i\in[s]}{\exists} p_{t_i} \underset{i\in[s]}{\exists} p_{\neg t_i} \bigwedge_{i\in[n]} \xi(C_i) \wedge \bigwedge_{i\in[s]} \tau(t_i)$$

Claim: θ and φ are logically equivalent.

- $\theta \vDash \varphi$: Suppose $I \vDash \theta$. We choose each h_i and g_t as the disjunction. Next, if $b = 0$, simply set all p_ℓ to zero. In turn, if $b = 1$, set p_ℓ to true if and only if $I \vDash \ell$. It is easy to check that this satisfies all $\xi(C_i)$ and $\tau(t_i)$. In particular, for each $h_i(p_{\ell_1^i}, \cdots, p_{\ell_r^i})$ there is $k \in [r]$ such that ℓ_k^i and hence $p_{\ell_k^i}$ must be true, as $I \vDash C_i$ by assumption.
- $\varphi \vDash \theta$: Suppose $I \vDash \varphi$. Then h_i and g_{t_i} are interpreted by some Boolean functions, and the p_ℓ by some truth values depending on b, such that all clauses in φ are true. In the case $b = 0$, the $h_i(\cdots)$ and $g_t(\cdots)$ must be false, and the same holds for all their arguments as well, due to the implications in $\xi(\cdots)$ and $\tau(\cdots)$. So $h_i(0, \ldots, 0) = g_t(0, 0) = 0$. In turn, in the case $b = 1$ it holds that $h_i(\cdots) = g_t(\cdots) = 1$, and hence at least one argument of each must have toggled its value. As a consequence, $\tau(t)$ forces that either p_t or $p_{\neg t}$ is true for every term t, and that p_ℓ is true iff $I \vDash \ell$. Likewise, for each $h_i(p_{\ell_1^i}, \ldots, p_{\ell_r^i})$ there is $k \in [r]$ such that $p_{\ell_k^i}$ and hence ℓ_k^i is true. In other words, all original clauses of θ are true in I.

The cases where θ contains quantifiers is proved analogously: b and the propositions $p_{t_i}, p_{\neg t_i}$ need to be quantified last in the prefix, as they depend on all other variables occurring in the formula, while the h_i and g_{t_i} are quantified first, since they can always just be set to the Boolean disjunction. Hence, the above proof works for arbitrary quantifier sequences $\mathcal{Q}$ and produces formulae of the form $\exists \vec{f} \mathcal{Q} \forall \vec{y} \exists \vec{z}$ as stated in the lemma. ◀

D Proof of Theorem 30

▶ **Theorem 30.** *Truth of formulae in* Σ_1^{sh} *is* **EXP**-*complete.*

Proof. The upper bound is given by Theorem 20. For the lower bound, we modify the proof of Theorem 29 and encode all reachable configurations of an **EXP** computation using a single function variable f. However, since the computation can use exponential space, f now takes a tape *address*, rather than the whole tape content, as well as a current timestep in binary as an argument.

Let M be a single-tape TM that decides an **EXP**-complete problem, where M has states Q, initial state q_0, accepting state q_f, rejecting state q_r, tape alphabet Γ, and transition relation δ. This time, we consider as a configuration a word over $\Gamma' := \Gamma \cup (Q \times \Gamma)$. For example, $a(q,b)c$ means that the machine currently is in state q and reads b at tape position two. Suppose M runs in time $2^{p(n)}$ for some polynomial p, $p(n) \geq n$, and uses the tape positions $\{1, \ldots, 2^{p(n)} - 2\}$. For technical reasons, we "pad" configurations with blank symbols $\Box$ at positions 0 and $2^{p(n)} - 1$, but these cells will never be visited. Let $\langle \cdot \rangle : \Gamma' \to \{0,1\}^k$ be some fixed encoding. The function f is now of arity $k + kp(n) + kp(n)$. The intended meaning of $f(\langle \alpha \rangle ; \mathsf{bin}(i); \mathsf{bin}(j))$ is that the ith symbol of the configuration on timestep j is α.

Let $x = x_1 \cdots x_n$ be the input. Let ℓ be minimal such that $n < 2^\ell$. We describe in the following formula that the first 2^ℓ symbols of the initial configuration are $\Box(q_0, x_1)x_2 \cdots x_n \Box \cdots \Box$ at timestep 0:

$$\psi_1 := f(\langle \Box \rangle ; \mathsf{bin}(0); \mathsf{bin}(0)) \wedge f(\langle (q_0, x_1) \rangle ; \mathsf{bin}(0); \mathsf{bin}(1)) \\ \wedge \bigwedge_{i=2}^{n} f(\langle x_i \rangle ; \mathsf{bin}(0); \mathsf{bin}(i)) \wedge \bigwedge_{i=n+1}^{2^\ell - 1} f(\langle \Box \rangle ; \mathsf{bin}(0); \mathsf{bin}(i))$$

Then the next formula also fixes the remaining blank symbols $\Box$ on tape positions from 2^ℓ to $2^{p(n)} - 1$.

$$\psi_2 := \forall \vec{v} \bigwedge_{j=1}^{p(n)-\ell} f(\langle \Box \rangle ; \mathsf{bin}(0); v_1, \ldots, v_{j-1}, 1, v_{j+1}, \ldots, v_{p(n)})$$

This is done by the third part of the arguments of f ranging over all numbers that have at least one of the first $p(n) - \ell$ bits set, which are $\{2^\ell, 2^\ell + 1, \ldots, 2^{p(n)} - 1\}$.

Next, we again state that M's rejecting configuration is *not* visited:

$$\psi_3 := \forall \vec{t} \forall \vec{u} \neg f(\langle (q_r, \Box) \rangle ; \vec{t}; \vec{u})$$

Finally, it remains to express in formulae that f is closed under transitions of M. As in Theorem 29, this is the only part of the formula where we introduce non-unit clauses, which now will rather be Horn instead of core. For this, we use another function variable suc ("successor"), which has arity $2p(n)$, and for which every term of the form $\mathsf{suc}(\mathsf{bin}(m), \mathsf{bin}(m+1))$ is true. We show how to enforce this later; for now, we use it to impose the aforementioned closure condition on f.

We consider the set of valid *windows* of M. A window is a sixtuple $(a_1a_2a_3; a'_1a'_2a'_3) \in (\Gamma')^6$. For example, $(a(q,b)c; ad(q,c))$ means that M in state q when reading b writes d and moves to the right. Cells not currently visited by the head do not change (except for the head moving onto a cell), so $(abc; abc)$ and $(abc; (q,a)bc)$ are valid windows but $(abc; abd)$ is not. The set W of valid windows is finite and only depends on the transition function of M. The following formula states that, whenever $(a_1a_2a_3; a'_1a'_2a'_3)$ is a valid window, the middle tape cell must become (or stay) a'_2.

$$\psi_4 := \forall \vec{t}\vec{s}\vec{u}\vec{v}\vec{w} \bigwedge_{(a_1a_2a_3;a'_1a'_2a'_3)\in W} \Big(\big(\mathsf{suc}(\vec{t};\vec{s}) \land \mathsf{suc}(\vec{u};\vec{v}) \land \mathsf{suc}(\vec{v};\vec{w}) \\ \land f(\langle a_1\rangle;\vec{t};\vec{u}) \land f(\langle a_2\rangle;\vec{t};\vec{v}) \land f(\langle a_3\rangle;\vec{t};\vec{w})\big) \to f(\langle a'_2\rangle;\vec{s};\vec{v})\Big)$$

Here, $\vec{t}$ and $\vec{s}$ encode consecutive timesteps, and $\vec{u}\vec{v}\vec{w}$ are adjacent positions. The first and last position must be separately fixed to $\Box$ because they are never in the middle of a window:

$$\psi_5 := \forall \vec{t}\big(f(\langle\Box\rangle;\vec{t};\langle 0\rangle) \land f(\langle\Box\rangle;\vec{t};\left\langle 2^{p(n)}-1\right\rangle)\big)$$

Next, we specify suc and finish the reduction:

$$\varphi := \exists \mathsf{suc}\, \exists f \left(\Big(\forall \vec{v} \bigwedge_{i=0}^{p(n)-1} \mathsf{suc}(v_1,\ldots,v_i,0,1^{p(n)-i-1}; v_1,\ldots,v_i,1,0^{p(n)-i-1})\Big) \land \bigwedge_{i=1}^{5} \psi_i \right)$$

Note that, just like f, the relation encoded by suc might contain more tuples than necessary, but again this does not hurt the reduction. It is easy to see that the formula can be transformed into a Σ_1 formula with simple matrix in Horn CNF. The Horn property of the formula hinges on ψ_4, for which it is crucial that M is deterministic. For this reason, this reduction cannot be generalized to, say, **NEXP**. ◀