V-HORN: A HORN-BASED SECOND-ORDER THEORY OF ARITHMETIC

by

Antonina Kolokolova

A thesis submitted in conformity with the requirements
for the degree of Masters
Graduate Department of Computer Science
University of Toronto

Abstract

V-Horn: a Horn-based second-order theory of arithmetic

Antonina Kolokolova

Masters

Graduate Department of Computer Science

University of Toronto

2000

In this thesis we present a second-order theory of arithmetic V-Horn, which encompasses polytime reasoning. It is equivalent in power to Zambella's P-def and Cook's QPV or PV_1; however, whereas those systems contain function symbols for all polytime functions, V-Horn achieves the same power by restricting the comprehension scheme to SO-Horn (as defined by Grädel) formulae. Our theory is possibly weaker than V_1^1 and, equivalently, S_2^1, in that it does not necessarily prove their comprehension (or induction) schemes.

We describe how to encode a run of Horn-SAT algorithm on a SO-Horn formula by a $SO\exists$-Horn formula, and use this result to demonstrate that SO-Horn is provably closed under complementation. The same method can be used to show that V-Horn provably defines all polytime functions.

We also show that some descriptive complexity results about SO-Horn logic, in particular the collapse of SO-Horn to its existential fragment, are formalizable in V_1-Horn, a restriction of V-Horn.

Acknowledgements

I have been very much honored to be supervised by Steve Cook, a most eminent scientist and an extremely nice person. His constant encouragement and unfailing support are invaluable. Whenever I thought my research hit the dead end, he would always suggest a right direction. I am looking forward to working towards my PhD under his guidance.

I am greatly indebted to Toni Pitassi, who sparkled my interest in the area of complexity theory, and guided my first steps while I was an undergraduate student. Without her, I probably would not be here now.

The department of Computer Science at UofT, especially the Theory Group, provide a truly exemplary research environment; its atmosphere of collaboration was very beneficial for my research. I am especially thankful to Valentine Kabanets, discussions with whom were immensely helpful to me in all stages of this work. I am also grateful to Oliver Kullman for referring me to the results that suggested the topic for this thesis.

Last but not least, I would like to thank Alasdair Urquhart for being a second reader for this thesis. His comments helped to improve it considerably.

Contents

Chapter 1

Introduction

In this chapter, we outline historical developments in bounded arithmetic and descriptive complexity, and explain how combining the two perspectives offered by these disciplines led to our main result — the construction of the second-order theory of arithmetic that captures polytime without explicitly introducing function symbols for all polytime functions.

1.1 Bounded Arithmetic

Since the celebrated results of Gödel on the power of Peano Arithmetic, there has been an increased interest in studying the power of weaker, bounded fragments of Peano Arithmetic. A further impetus has been provided by the development of computer science: the notion of feasible computation motivated the study of theories corresponding to complexity classes below EXPTIME. Since the complexity classes P and NP and relation between them attract most of the attention in complexity theory, it is natural that there is an interest in studying the corresponding fragments of Peano Arithmetic.

Cobham [Cob65] was the first to give a machine-independent description of the class of polytime functions P. He showed that the class of polytime functions is the smallest class closed under composition and limited recursion on notation; this allowed the treatment of polynomial functions in the context of pure logic.

In 1971, Parikh [Par71] proposed the first system of bounded arithmetic, called $I\Delta_0$, where the induction scheme was restricted to bounded (Δ_0) formulae. He showed that all functions that are Δ_0-definable in $I\Delta_0$ are polynomially bounded; i.e., if ϕ is a Δ_0 formula and $I\Delta_0 \vdash \forall\bar{x}\exists y\phi(\bar{x}, y)$, then the value of y is bounded by a polynomial in $\bar{x}$. However, $I\Delta_0$ does not allow some important properties, in particular coding of polynomial length proofs. To allow for that, Paris and Wilkie [PW81a, PW81b] later extended $I\Delta_0$ by adding the axiom Ω_1, stating the totality of the function $x^{|x|}$.

The first theory that was explicitly designed in order for all proofs to be feasibly constructible (i.e., constructible in polynomial time) was the equational theory PV, proposed by Cook in 1975 [Coo75]. There, Cobham's characterization of polytime was used to construct polytime functions. One motivation for PV was its close relation with the Extended Frege proof system: theorems of PV correspond to families of tautologies with polynomial length proofs. Another candidate for the system with feasibly constructive proofs was the intuitionistic version of PV, IPV, presented in [CU93]. This system, as well as its classical version CPV, includes an induction on NP predicates; that makes CPV equivalent in power to V_1^1 and S_2^1.

Later, Cook [Coo98b] developed a quantified version of PV called QPV, and used it to study the relationship between NC_1 and P from the point of view of corresponding theories, where $QALV$ represents NC_1 in the same sense as QPV represents P. Since PV includes the induction of notation, the system QPV, axiomatized by the universal closures of the theorems of PV, has enough power for polytime reasoning; however, it is possibly weaker than CPV in that it might not prove the induction on NP predicates from CPV.

The major work establishing the relation of complexity theory and bounded arithmetic was the 1986 PhD thesis of S.Buss [Bus86], where a number of first-order and second-order theories that characterize the polytime hierarchy (starting with the first level), PSPACE and EXPTIME were developed. The most important of them is the first-order theory S_2^1, consisting of a set of 32 axioms and a $PIND$ induction scheme (induction on Σ_1^b formulae). Buss proves that predicates are polynomial iff S_2^1 proves that they are in NP$\cap$co-NP. He also shows that S_2^1 is Σ_1^b conservative over PV; however, general conservativity of S_2^1 over QPV would imply collapse of polynomial hierarchy to P/poly [Bus95, KPT91, Zam96]. Razborov and, at the same time, Takeuti [Raz93, Tak93] show the equivalence between first-order and second-order hierarchies, which can be used to show that V_1 (V_1^1 in Krajiček's notation) is equivalent in power to S_2^1 [Raz93].

Finally, in 1996 Zambella [Zam96] introduced a hierarchy of second-order theories P_i-def, and studied its relation with the hierarchy BA of theories of Σ_i^p-comp, $i \geq 1$. The lowest level of P_i-def hierarchy, after the AC_0 level, is the theory P-def: a second-order theory equivalent to QPV in power. In his paper Zambella proves several interesting results, in particular that if P_i-def $\vdash \Sigma_{i+1}^p$-comp, then polytime hierarchy provably collapses to Σ_{i+3}^p. Similar result was obtained independently by Buss [Bus95].

1.2 Descriptive Complexity

While in bounded arithmetic we are interested in the proving power of theories, in descriptive complexity we are interested in expressive power of logics. Instead of asking what class of theorems can be proven, we ask what properties (in particular, graph properties) we can express in certain logics.

This direction of research started in 1974 with Fagin's work on finite model theory [Fag74]. Fagin noticed that for each fixed second-order formula, the set of its finite models forms a language in NP, and every language in NP can be encoded in this fashion. That is, we can verify in polynomial time if a certain structure is a model of a formula, and, for the other direction, we can represent a run of an NP Turing machine by a second-order existential formula [Fag93].

Later, this result was extended by Stockmeyer [Sto77] to the equivalence between the polytime hierarchy and second-order formulae. However, it was much harder to characterize the class P: all existing characterizations require additional restrictions, namely the presence of successor relation in the language. The monadic fragment of second-order existential logic appears too weak: it does not include even some problems in P, such as graph connectivity or reachability [FSV95].

When attempts to capture P by restricting second-order logic failed, researchers tried to create a suitable characterization by augmenting first-order logic. It was proven by Barrington, Immerman and Straubing [BIS90] that first-order logic with BIT predicate has expressive power equivalent to uniform AC_0; therefore, a stronger logic than just first-order was needed. One such characterization of P uses the first-order logic augmented with the least fixed point operator [Var86, Imm86]. It also requires presence of a successor relation in the language.

In 1991, Grädel came up with a second-order logic that captures P in the presence of successor in the language [Grä91]. He restricted second-order logic to $SO\exists$-Horn formulae, using the fact that Horn-SAT can be done in polynomial time. He then proved that this $SO\exists$-Horn logic is equivalent to first-order logic with the least fixed point operator in the presence of successor, and hence $SO\exists$-Horn captures P. The equivalence between $SO\exists$-Horn and P follows almost immediately from the proof of Fagin's theorem (see, e.g., [SP98]).

1.3 Our Results

The idea of this thesis came from studying the representations of complexity classes by bounded arithmetic and by logics in descriptive complexity. There seems to be an interesting relation between the power of a logic and the power of a second-order theory with comprehension scheme restricted to formulae from that logic. Let Σ_i^B formulae be second-order formulae where all quantifiers are bounded and there are at most i levels of alternating second-order quantifiers. Then, Σ_i^B formulae, $i \geq 1$, which correspond to the levels of polytime hierarchy by Stockmeyer's result, give Zambella's BA when used in the comprehension scheme. The equivalence between uniform AC_0 and first-order logic+BIT, proven by Barrington, Immerman and Straubing in [BIS90], similarly resembles the relation between V_0 and AC_0; moreover, in that case the class of functions definable in V_0 is AC_0 functions. Extending this analogy, we build a second-order theory with comprehension restricted to $SO\exists$-Horn formulae. That is, the objects (sets) that we can construct are definable by $SO\exists$-Horn formulae, and hence are polytime constructible. This allows us to construct a theory with the power of P-def without introducing any function symbols, which seems simpler and somewhat more natural.

First, we show that our theory V_1-Horn is an extension of V_0. That is, we can define all AC_0 relations in V_1-Horn. Later, it can be used for the proof of equivalence of V_1-Horn with P-def: V_1-Horn can define all AC_0 functions (Zambella's "rudimentary functions"). Since Σ_0^B formulae can contain first-order existential quantifiers, which are not allowed in $SO\exists$-Horn formulae, we convert Σ_0^B formulae into provably equivalent $SO\exists$-Horn formulae.

The first of our two main results is that the collapse of $SO\exists$-Horn to SO-Horn is provable in V_1-Horn. We show that Grädel's proof of the collapse is formalizable in V_1-Horn. In general, it is interesting to check how results about descriptive power of logics can be formalized in the corresponding theories of arithmetic. It is especially true about the collapse results in light of the results of Zambella [Zam96], Buss [Bus95] and Krajiček, Pudlak and Takeuti [KPT91], which relate the provable collapse of polytime hierarchy and the collapse of boolean hierarchy.

Finally, we show how to encode a run of a $SO\exists$-Horn-SAT algorithm by a $SO\exists$-Horn formula. This technique is used to show the provable closure of the class of $SO\exists$-Horn formulae under complementation and first-order quantification. It can also be useful in defining a class of functions that is provably closed under composition and limited recursion, which, in turn, is needed to show that V_1-Horn is at least as powerful as P-def.

Chapter 2

Preliminaries

2.1 Horn Formulae

Definition 2.1. *A propositional Horn formula is a CNF formula, such that at most one atom occurs positively in each of its clauses. Equivalently, we can view such clauses as having the form* $(l_1 \wedge l_2 \ldots \wedge l_{k-1} \to l_k)$*, where* l_i *are positive literals.*

Definition 2.2. *A second-order Horn (SO-Horn) formula (over some first-order language) in canonical form is*

$$Q_k P_k \ldots Q_1 P_1 \forall x_m \ldots \forall x_1 \phi,$$

where Q_i *are* $\exists$ *or* $\forall$ *quantifiers,* $P_k \ldots P_1$ *are unary relation symbols,* $x_m \ldots x_1$ *are first-order variables and* ϕ *is a quantifier-free CNF formula, in which every clause contains no more than one atom of the form* $\neg P_i(t)$ *for some term* t*. If all* Q_i *are existential quantifiers, we say that it is a second-order existential Horn formula (SO*$\exists$*-Horn).*

Every clause can be equivalently represented as $l_1 \wedge \ldots \wedge l_{n-1} \to l_n$*, where none of* l_i *can be of the form* $\neg P_i(t)$*. We say that* ϕ *is a Horn formula with respect to the quantified second-order variables. There are no occurrences of* $|P_i|$ *in* ϕ*.*

It is important to have only universal first-order quantifiers, since for structures of cardinality at least 2, any second-order existential formula is equivalent to a formula of the form $\exists P_1 \dots P_n \forall \bar{y} \exists \bar{z} \phi$, where ϕ is a conjunction of Horn clauses. For structures of cardinality 3, every second-order formula is equivalent to a second-order formula with both universal and existential second-order quantifiers and first order part of the form $\exists \bar{y} \forall \bar{z} \phi$, where ϕ is a conjunction of Horn clauses. [Grä91]

Since we use $SO\exists$-Horn formulae in the context of bounded arithmetic, every first-order quantifier in a $SO\exists$-Horn formula is bounded by a polynomial in free variables. An additional restriction is that we do not allow the upper bound of a quantified variable ($|P_i|$) as a part of a $SO\exists$-Horn formula. Intuitively, we may be able to check for existence of a number i in a set by asking if the least upper bound of the set is larger than i.

Note that only the conjunction of Horn formulae is a Horn formula, but not necessarily the disjunction or negation.

Example 2.1 *(Parity is in $SO\exists$-Horn).* Parity of a string X can be expressed by a formula

$$\exists P_{even} \exists P_{odd} \forall n < |X| \phi,$$

where $\phi \equiv \alpha \wedge \beta \wedge \eta \wedge \zeta$ is a first-order Horn formula. Here,

$$\alpha \equiv (X(0) \rightarrow P_{odd}(0)) \wedge (\neg X(0) \rightarrow P_{even}(0)).$$

states that $P_{odd}(0)$ is set if the first bit of X is 1, and $P_{even}(0)$ is set otherwise. The second condition is

$$\beta \equiv (\neg P_{even}(n) \vee \neg P_{odd}(n)).$$

It states that $P_{even}(n)$ and $P_{odd}(n)$ cannot both hold. Together, α and β imply that exactly one of $P_{even}(0)$ and $P_{odd}(0)$ holds, thus expressing the parity of the first bit of X. The inductive step is stated by

$$\begin{aligned} \eta \equiv &(X(n+1) \wedge P_{even}(n) \rightarrow P_{odd}(n+1)) \wedge (X(n+1) \wedge P_{odd}(n) \rightarrow P_{even}(n+1)) \\ &\wedge (\neg X(n+1) \wedge P_{even}(n) \rightarrow P_{even}(n+1)) \wedge (\neg X(n+1) \wedge P_{odd}(n) \rightarrow P_{odd}(n+1)). \end{aligned}$$

It says that if the current bit of X is set to 1, then the two predicates change their values. The new values are still opposite: at least one predicate will be set because of α, and the other will have the opposite value because of β. If the current bit of X is 0, the (opposite) values are preserved.

The end condition is

$$\zeta \equiv P_{odd}(|X|).$$

Note that it is not necessary to state explicitly $P_{even}(n) \vee P_{odd}(n)$ (which is not a Horn clause). This condition is implicitly given by α and η.

2.2 Second-Order Theories

The language of weak second-order logic consists of function and predicate symbols, logical connectives and quantifiers, together with two sorts of variables: first-order and second-order. Capital letters are used to denote the second-order variables, and lowercase letters the first-order variables.

In the standard model, the first-order variables are interpreted as numbers (indices), and the second-order variables as finite sets of numbers. The second-order objects can also be viewed as numbers in binary; then the first-order objects correspond to numbers in unary.

For our second-order theory V_1-Horn, we will use the language of Peano Arithmetic together with the set membership relation $\in$ ($x \in X$ can be also written as $X(x)$) and the strict upper bound operation $|\ |$

$$\mathcal{L}_A^2 = \{0, 1, +, \cdot, |\ |; \in, \leq, =\}$$

Here, $+$ and $\cdot$ are first-order operations, and $\leq$ and $=$ first-order relations, with the usual meanings. The relation $\in$ and the operation $||$ relate first-order and second-order objects: $x \in Y$ holds iff x is in the set Y, the operation $|X|$ takes an element X and returns a first-order object equal to the largest element of X plus 1, or 0 if X is empty.

Strings can be represented as nonempty sets, by considering the last element in the set to be the "delimiter" of a string. Thus, for a nonempty set X, $X(i) = 1$ will correspond to $i \in X$, and the length of a string will be the largest element in X, $|X| - 1$. An empty string corresponds to a set containing only 0. An empty set does not correspond to any string.

2.3 Axioms of V_1-Horn

We define V_1-Horn to be a second-order theory with the following axioms:

1. Axioms of Peano Arithmetic:

$$\begin{aligned}
\text{B1} &: x + 1 \neq 0 \\
\text{B2} &: x + 1 = y + 1 \rightarrow x = y \\
\text{B3} &: x + 0 = x \\
\text{B4} &: x + (y + 1) = (x + y) + 1 \\
\text{B5} &: x \cdot 0 = 0 \\
\text{B6} &: x \cdot (y + 1) = (x \cdot y) + x
\end{aligned}$$

2. Axioms for $\leq$:

$$\begin{aligned}
\text{B7} &: x \leq x + y \\
\text{B8} &: (x \leq y \wedge y \leq x) \rightarrow x = y \\
\text{B9} &: 0 \leq x \\
\text{B10} &: x \leq y \leftrightarrow x < y + 1
\end{aligned}$$

3. Additional axioms (existence of predecessor and transitivity of $\leq$):

$$\text{B11} : x \neq 0 \rightarrow \exists y(y + 1 = x)$$

$$\text{B12} : x \leq y \wedge y \leq z \rightarrow x \leq z$$

4. Length axioms:

$$\text{L1} : X(y) \rightarrow y < |X|$$

$$\text{L2} : y + 1 = |X| \rightarrow X(y)$$

We also need a comprehension scheme. In V_1-Horn, we will use

$$\Sigma_1^B\text{-Horn-COMP} : \exists X \forall z < y(z \in X \leftrightarrow \phi(z)),$$

where $\phi(z)$ is any second-order existential Horn formula not containing X, and y is a free variable.

In V-Horn, the comprehension scheme will be

$$\Sigma^B\text{-Horn-COMP} : \exists X \forall z < y(z \in X \leftrightarrow \phi(z)),$$

where $\phi(z)$ is any second-order Horn formula not containing X and y is a free variable.

All second-order objects in V_1-Horn have a finite length. However, we usually do not state the bound on the length explicitly. Since all first-order quantifiers are explicitly bounded by a polynomial in the free variables, and there are no applications of $||$ in SO-Horn or $SO\exists$-Horn formulae, only a finite part of the second-order variable can be referenced. We assume that the length of the set is the strict upper bound on the indices, and no unreachable index is in the set. That is, the comprehension always gives the *smallest* set satisfying the condition.

2.4 Versions of Induction

There is no induction axiom among the axioms of V_1-Horn. However, induction can be derived as a theorem. First, we can derive the least number principle (LNP) from the length axioms; then, the induction scheme can be proven using LNP.

Lemma 2.3. *The least number principle is a theorem of V_1-Horn.*

$$\textit{LNP: } |X| > 0 \rightarrow \exists x < |X|(X(x) \wedge \forall y < x \neg X(y))$$

Proof. We apply comprehension to obtain a set Y, such that $z \in Y \leftrightarrow \forall i < |X|(i \in X \rightarrow z < i)$. This is a $SO\exists$-Horn formula, since X is a free variable. Set Y contains all elements that are smaller than all elements of X, in particular smaller than the minimal element of X. Since X is nonempty, there is at least one element in X (since by $B11$, $|X| > 0 \rightarrow \exists x(|X| = x + 1)$, and by $L2$, $x + 1 = |X| \rightarrow X(x)$).

Now we need to show that $|Y| < |X|$. Suppose $|Y| \geq |X|$. By $B12, 1 \leq |X| \wedge |X| \leq |Y| \rightarrow 1 \leq |Y|$. Then, by $B10, 0 < |Y|$ and so using $L2$ and $B11$ as above, $|Y| > 0 \rightarrow \exists y(|Y| = y+1)$ and $y+1 = |Y| \rightarrow Y(y)$. Then by $B12$ and $B10$ we get $y \geq x$, where $X(x)$ and $Y(y)$. But by definition of Y, $\forall y < |X| \forall x < |X|(x \in X \rightarrow y < x)$. Therefore, $|Y| < |X|$.

So, the first element that is not in Y is the smallest element of X (by $B12$). By length axioms, if a string $|Y|$ is nonempty, then the predecessor y of $|Y|$, which exists by $B11$, is a maximal element of $|Y|$: by $L1$, $B12$ and $B10$, all other elements are at most as large as y, and by $L2$, y is in the set. Therefore, $|Y|$ is the minimal element of X. □

Lemma 2.4. *Induction on length of a string is a theorem of V_1-Horn.*

$$\textit{IND: } (X(0) \wedge \forall y < z(X(y) \rightarrow X(y + 1))) \rightarrow X(z)$$

Proof. We can prove the contrapositive: $\neg IND \rightarrow \neg LNP$. Suppose that for a set X, both

$$X(0) \wedge \forall y < z(X(y) \rightarrow X(y + 1)))$$

and $\neg X(z)$ hold. Our comprehension scheme Σ_1^B-Horn-COMP guarantees the existence of a set Y, which corresponds to the complement of X, with $z+1$ as a bound on length of Y. Since $z \notin X$, we have $|Y| = z+1$. Thus, $|Y| > 0$, so if LNP holds, then there exists a minimal element x in Y.Then, $\forall w < xX(w) \wedge \neg X(x)$. If $x = 0$, then this condition violates $X(0)$. Otherwise, $0 \leq x-1 < z$. In this case, $X(x-1)$ holds, but $X(x)$ does not, and that contradicts $\forall y < z(X(y) \rightarrow X(y+1))$ for the case $y = x-1$. Therefore, LNP does not hold. Thus, $LNP \rightarrow IND$. □

Lemma 2.5. *Induction on length of a string with an arbitrary basis is a theorem of V_1-Horn. That is, if X is a string, and i is a free variable,*

$$V_1\text{-}Horn \vdash (\forall y < a(i < a \wedge X(i) \wedge X(y) \rightarrow X(y+1))) \rightarrow X(a)$$

Proof. We can use Σ_1^B-Horn-COMP with $\phi(z) \equiv X(z+i)$, and bound b, for which $b+i < a$. Now, $X(i) \rightarrow Z(0)$ and $\forall y < a(X(x) \rightarrow X(y+1))$ implies $\forall y+i < a(X(y+i) \rightarrow X(y+i+1))$, which in turn gives $\forall z+i < a(Z(z) \rightarrow Z(z+1))$. By applying induction to Z we get $Z(b)$. Since $X(a) \leftrightarrow Z(b)$, $X(a)$ holds. □

2.5 Operations on Second-Order Objects

Minimization function. Minimization function denotes the smallest element in the set (less than the length). If the set is empty, the length is returned. Thus, z is the minimal element in X iff

$$\forall y < zX(z) \wedge (y \notin X)$$

Pairing function. We encode a pair $< x, y >$ by z, where $2z = (x+y)(x+y+1)+2x$. This function is a bijection.

Array. An array X is represented as a set of pairs $< i, j >$, for which the element of the array is 1. A row i of X is denoted $X^{[i]}$, and is a set consisting of all pairs with first element equal to i.

Set operations. Set operations such as $X \cup Y$ and $X \cap Y$ can be defined using comprehension with $\phi \equiv X(z) \wedge Y(z)$ and $\phi \equiv X(z) \vee Y(z)$.

Since in all definitions above there are no quantified variables, they can be considered V_1-Horn formulae.

2.6 Other Second-Order Theories.

To state some of the results, we need definitions of three other second-order theories. The descriptions of all three of them are based on the definitions by Zambella in [Zam96]. Definitions of V_0 and V_1 are cited from [Coo98a], the definition of P-def directly from [Zam96].

2.6.1 Second-Order Theories V_0 and V_1.

Definition 2.6. *We define, following the notation in [Coo98a] a Σ_i^B formula to be a second-order formula with bounded first-order and second-order quantifiers, which contains at most i alternations of bounded second-order quantifiers, starting with existential quantifiers. That is, there are only existential bounded second-order quantifiers in Σ_1^B formulae, and no second-order quantifiers in Σ_0^B.*

Since the description of V_1-Horn was based on the descriptions of V_0 and V_1, these three theories are very similar. Their language and the first 10 axioms are the same. However, V_0 and V_1 do not have B11 and B12 as axioms, and instead of L2 they include IND axiom scheme. The comprehension is on Σ_0^B formulae in V_0 and on Σ_1^B formulae in V_1. Also, in V_1-Horn the bounds on second-order variables are not explicitly specified, whereas in V_0 and V_1, as well as in P-def, they are. Moreover, in V_0 and V_1 the second-order variables are interpreted as strings, not sets, and so the value of the length function cannot be directly obtained from the values of the indices.

As shown in [Coo98a], the functions definable in V_0 are AC_0 functions, and functions definable in V_1 are polytime functions.

Note that as it was shown earlier, IND is a theorem of V_1-Horn, and Σ_1^B-Horn-COMP is just a restriction of Σ_1^B-COMP. In [Kra95], V_0 and V_1 correspond to, respectively, V_1^0 and V_1^1; Zambella refers to them as "theory of Σ_0^p-comp" and "theory of Σ_1^p-comp".

2.6.2 Second-Order Theory P-def.

The axioms of P-def are:

$$
\begin{array}{ll}
0 \neq 1 & a \cdot (b+1) = (a \cdot b) + a \\
a + 0 = a & a \leq b \leftrightarrow a < b + 1 \\
a + 1 = b + 1 \rightarrow a = b & a + 1 \leq b \leftrightarrow a < b \\
a + (b+1) = (a+b) + 1 & A < b \leftrightarrow (\forall x \in A) x < b \\
a \neq 0 \leftrightarrow (\exists x < a) x + 1 = a & A = B \leftrightarrow A \subseteq B \wedge B \subseteq A \\
A \cdot 0 = 0 & A \neq \emptyset \rightarrow (\exists x \in A)(A < x + 1)
\end{array}
$$

In P-def, Σ_0^B comprehension scheme is used. The greater power than that of V_0 is given to it by adding function symbols for all polytime computable string and number functions. The polytime functions are defined as a closure of AC_0 functions (ones obtained using Σ_0^B-COMP and minimization on these) under limited recursion and composition.

Chapter 3

V_1-Horn Is an Extension of V_0

We can show that, given an AC_0 relation, we can represent it as a $SO\exists$-Horn formula. That implies that V_1-Horn contains V_0. It is a proper inclusion, since parity, which is not an AC_0 relation, can be represented in $SO\exists$-Horn.

3.1 Changing the Order of Quantifiers in $SO\exists$-Horn.

Lemma 3.1 (Quantifier Exchange rule).

$$V_1\text{-}Horn \vdash \forall y < b \exists X'_k \dots \exists X'_1 \phi(\bar{X}', y) \leftrightarrow \exists X_k \dots \exists X_1 \forall y < b \phi(X_k^{[y]}, \dots, X_1^{[y]}, y),$$

where ϕ is a Horn formula with respect to $X_1 \dots X_k$ with no second-order quantifiers. Note that the length of X_i is bounded by $c_i = \frac{1}{2}(a_i + b)(a_i + b + 1) + b$.

Proof. For clarity, we will consider only the case of $k = 1$. The case of $k = 0$ is trivial, and for $k > 1$ we just repeat the same reasoning for k variables X_i simultaneously.

1. $\exists X \forall y < b\phi(X^{[y]}, y) \rightarrow \forall y < b \exists X' \phi(X', y)$

   Let b be some fixed number. Suppose $\exists X \forall y < b\phi(X^{[y]}, y)$ holds. Since pairing function is a bijection, for every y it is possible to find a unique set X', containing i for which pairs of indices $< y, i >$ are in X. If there are no such i, X is empty;

$|X'| \leq |X|$. Such X' is the y^{th} row of X by the definition of a row, and so $\phi(X', y)$ holds iff $\phi(X^{[y]}, y)$ does. If $\phi(X^{[y]}, y)$ holds for some $y < b$, then $X^{[y]}$ is a witness for $\exists X' \phi(X', y)$. Since $\phi(X^{[y]}, y)$ holds for every $y < b$, $\forall y < b \exists X' \phi(X', y)$ holds as well.

2. $\forall y < b \exists X' \phi(X', y) \rightarrow \exists X \forall y < b \phi(X^{[y]}, y)$

   Since $\phi(X, y)$ does not contain any second-order quantifiers, $\psi \equiv \exists X \forall y < b \phi(X^{[y]}, y)$ is a $SO\exists$-Horn formula. Then, we can obtain a characteristic string Z of ψ from the comprehension axiom, and state the induction for Z (induction on b).

$$(Z(0) \wedge \forall b < z(Z(b) \rightarrow Z(b+1))) \rightarrow Z(z)$$

   Let $b = 0$, then $\forall y < 0 \psi(X, y)$ holds for any X and ψ. Therefore, $\exists X \forall y < 0 \phi(X, y)$ holds.

   Now we need to show $Z(b) \rightarrow Z(b+1)$.

   Here, $Z(b)$ gives us $\exists Y \forall y < b \phi(Y^{[y]}, y)$. By assumption that $\forall y < b+1 \exists Y' \phi(Y', y)$, $\exists Y' \phi(Y', b)$. Since pairing function is a bijection, sets Y and Y' are disjoint. Define X to be $X = Y \cup Y'$. Since it contains exactly those pairs with $y < b$ as a first element that are contained in Y, and pairs with b as a first element that are contained in Y', $X^{[y]} = Y^{[y]}$ iff $y < b$, and $X^{[b]} = Y'$. Therefore, this X is a witness for $\exists X \forall y < b + 1 \phi(X^{[y]}, y)$, and thus $Z(b+1)$ holds. By induction, $Z(z)$ holds.

□

Corollary 3.2. *Every formula Ψ that consists of second-order $\exists$ and first-order bounded $\forall$ quantifiers (in any order), followed by a quantifier-free formula ϕ, which is Horn with respect to the second-order $\exists$ quantifiers, is provably equivalent to a $SO\exists$-Horn formula Ψ' in canonical form.*

Proof. We can prove this statement by induction on the number of FO quantifiers in the formula. That is, we will prove that if all formulae with at most k FO quantifiers can be converted to canonical form, then a formula with $k+1$ FO quantifiers can be converted to canonical form.

The basis is when there is only one FO quantifier, $\forall x$. Consider the subformula starting with it, $\psi \equiv \forall x < b \exists Y_1 \ldots \exists Y_m \phi(Y_1, \ldots, Y_m)$. By lemma 3.1, $\psi \equiv \psi' \equiv \exists Y_1' \ldots \exists Y_m' \forall x < b \phi(Y_1'(x), \ldots, Y_m'(x))$. By replacing ψ by ψ' in the original formula, we obtain an equivalent formula in the canonical form.

Suppose now that a formula contains $k+1$ FO quantifiers. Let $\forall x$ be the outermost FO quantifier. Consider the subformula $\psi \equiv \forall x \phi$. Here, ψ has k FO quantifiers, and so by induction hypothesis it can be converted to an equivalent formula in the canonical form ψ'. That is, $\psi' \equiv \exists Y_m \ldots \exists Y_1 \phi$, where ϕ does not contain any second-order quantifiers. Now we can apply lemma 3.1 again to move $\forall x$ inwards past $\exists Y_m \ldots \exists Y_1$. That gives us a $SO\exists$-Horn formula in the canonical form. □

We can now use the term $SO\exists$-Horn to refer to the formulae with mixed quantifiers of the form described in the corollary above.

3.2 Converting Σ_0^B Formulae to $SO\exists$-Horn

The restriction on the form of formulae for Σ_1^B-Horn comprehension scheme does not allow us to apply Σ_1^B-Horn-COMP to first-order and Σ_0^B formulae directly. The problem arises because of the existential first-order quantifiers, which are not allowed in the $SO\exists$-Horn formulae. However, it is possible to create an equivalent $SO\exists$-Horn formula, using the following construction. The idea is to simulate a quantified first-order variable x by a universally quantified first-order y and two second-order variables, P_x and $\tilde{P}_x$. This construction is repeated for all first-order quantifiers.

Example 3.1. In the simplest case, consider a Σ_0^B formula $\psi \equiv \exists x < a \phi(x)$, where ϕ is first-order quantifier-free with string and number free variables. Since there are no string quantifiers in ψ, $\phi(x)$ can safely be negated (input variables are negatable). The corresponding $SO\exists$-Horn formula is

$$\Psi \equiv \exists P \exists \tilde{P} \forall x < a \phi'(P, \tilde{P}, x),$$

where $\phi' \equiv \alpha \wedge \beta \wedge \eta \wedge \zeta$ is first-order Horn formula, consisting of

- Initialization:

$$\alpha \equiv \tilde{P}(0) \wedge (\neg P(0) \vee \neg \tilde{P}(0))$$

- Restriction:

$$\beta \equiv (\neg P(x+1) \vee \neg \tilde{P}(x+1))$$

- Transition and propagation:

$$\eta \equiv (\phi(x) \to P(x+1)) \wedge (\neg \phi(x) \wedge \tilde{P}(x) \to \tilde{P}(x+1)) \wedge (P(x) \to P(x+1))$$

- End condition:

$$\zeta \equiv P(a)$$

Here, the length of P and $\tilde{P}$ is limited by $a+1$, since a is the largest referenced index: $P(a)$ and $\tilde{P}(a)$ contain the resulting values.

The role of the initialization is to set one of $P(0)$ and $\tilde{P}(0)$. The condition $(\neg P(0) \vee \neg \tilde{P}(0))$ forces at least one of the predicates to be false in 0, so exactly one of $P(0)$ and $\tilde{P}(0)$ is set. If, as in this case, the original quantifier is existential, then we are looking for a witness and initialize with $\tilde{P}(0) = 1$ (and, thus, $P(0) = 0$) to state that "no witness found yet". In the case of universal quantifier, we initialize with $P(0) = 1$ to state that "no counterexample found yet".

For any fixed index $x > 0$, transition and propagation force one of the two predicates to hold, and restriction forces at least one to be false, so that the predicates $P(x)$ and $\tilde{P}(x)$ always have the opposite values. If the witness is found now ($\phi(x)$ holds), or was found before ($P(x)$ holds), then $P(x+1)$ is set. Otherwise, the condition of the third clause holds, and $\tilde{P}(x+1)$ is set. Once $P(x)$ is set, it continues to hold for larger values of x, up to (and including) a, and thus $P(a)$ holds iff a witness was found. That fact is expressed as the end condition.

This example corresponds to the base case for converting a formula with alternating first-order quantifiers to $SO\exists$-Horn form. If the innermost quantifier is a universal quantifier, then α and η have to be changed slightly: in α, $\tilde{P}(0)$ becomes $P(0)$; η becomes

$$\eta \equiv (\neg\phi(x) \to \tilde{P}(x+1)) \wedge (\phi(x) \wedge P(x) \to P(x+1)) \wedge (\tilde{P}(x) \to \tilde{P}(x+1)).$$

The end condition is only necessary for the conversion of the outermost first-order quantifier; however, equivalents of other conditions are needed for the conversion of each preceding first-order quantifier as well.

After we have converted a subformula starting with i^{th} quantifier, we can check its truth value by checking which of P_i and $\tilde{P}_i$ holds with the bound for i^{th} variable as the last parameter. By construction exactly one of them holds; that allows us to check the value of the subformula without negating it. Thus, when we write the conditions for the i^{th} quantified variable, we only need access to the two predicates created for $i-1$-th quantifier. They play the role of ϕ and $\neg\phi$ of the base case.

Suppose we are converting i^{th} quantifier (counting from the innermost). At this moment, the formula is of the form

$$\begin{aligned}\Psi_{i-1} \equiv Q_k x_k < a_k \ldots Q_{i+1} x_{i+1} < a_{i+1} Q_i x_i < a_i \exists P_{i-1} \le a_{i-1} + 1 \exists \tilde{P}_{i-1} \le a_{i-1} + 1 \\ \ldots \exists P_1 \le a_{i-1} \ldots a_2(a_1+1) \exists \tilde{P}_1 \le a_{i-1} \ldots a_2(a_1+1) \forall x_{i-1} < a_{i-1} \ldots \forall x_1 < a_1 \phi_{i-1},\end{aligned} \tag{3.1}$$

where quantifiers $Q_k \ldots Q_i$ are the original first-order quantifiers. The order of the first-order quantifiers and the constructed second-order quantifiers corresponds to the order of the original quantifiers. [1]

[1] In this and some of the following formulae the bounds on the length of the second-order variables are stated explicitly for clarity: $\exists P \le a$ means that $\exists P(|P| \le a+1)$.

During the i^{th} step of the construction, we are concerned with the subformula starting with Q_i. It has $x_k \dots x_{i+1}$ as free variables. We are replacing $Q_i x_i$ by $\exists P_i \exists \tilde{P}_i \forall x_i$, then, using the Quantifier Exchange lemma, we move $\forall x_i$ inwards to place it after all second-order quantifiers. That introduces a new dimension to each of the arrays $P_{i-1}, \tilde{P}_{i-1}, \dots, P_1, \tilde{P}_1$. This way, when we finish our construction, P_1 and $\tilde{P}_1$ will have the number of dimensions equal to the number of the first-order quantifiers k; each pair of strings will have the number of dimension one less then the previous pair. The newly constructed P_i and $\tilde{P}_i$ have length $a_i + 1$, but when we increase the dimension, we only consider the possible values for the first-order variables, not including the bound. That is, after we move $\forall x_i$ inside past the second-order quantifiers, each of the previously constructed arrays receives one more dimension of length a_i. Thus, the two innermost string variables have length $a_i \dots a_2(a_1 + 1)$, the next two have length $a_i \dots a_3(a_2 + 1)$ and so on.

Now, we can construct the formula

$$\begin{aligned}
\Psi_i \equiv & Q_k x_k < a_k \dots Q_{i+1} x_{i+1} < a_{i+1} \\
& \exists P_i \leq a_i + 1 \exists \tilde{P}_i \leq a_i + 1 \\
& \exists P_{i-1} \leq a_i(a_{i-1} + 1) \exists \tilde{P}_{i-1} \leq a_i(a_{i-1} + 1) \\
& \quad \dots \\
& \exists P_1 \leq a_i \dots a_2(a_1 + 1) \exists \tilde{P}_1 \leq a_i \dots a_2(a_1 + 1) \\
& \quad \forall x_i < a_i \forall x_{i-1} < a_{i-1} \dots \forall x_1 < a_1 \phi_i.
\end{aligned}$$

The first-order part of Ψ_i will be $\phi_i = \phi_{i-1} \wedge \alpha_i \wedge \beta_i \wedge \eta_i$, where ϕ_{i-1} is the first-order quantifier-free part of the formula constructed on the previous step, and the last three subformulae of ϕ_i play the same roles as the corresponding subformulae of the base case:

- Initialization. If Q_i is an existential quantifier, then

$$\alpha_i \equiv \tilde{P}_i(0) \wedge (\neg P_i(0) \vee \neg \tilde{P}_i(0))$$

Otherwise, if Q_i is a universal quantifier

$$\alpha_i \equiv P_i(0) \wedge (\neg P_i(0) \vee \neg \tilde{P}_i(0))$$

- Restriction:

$$\beta_i \equiv (\neg P_i(x_i + 1) \vee \neg \tilde{P}_i(x_i + 1))$$

- Transition and propagation. If Q_i is an existential quantifier, then

$$\begin{aligned} \eta_i \equiv & (P_{i-1}(x_i, a_{i-1}) \rightarrow P_i(x_i + 1)) \wedge (P_i(x_i) \rightarrow P_i(x_i + 1)) \\ & \wedge (\tilde{P}_{i-1}(x_i, a_{i-1}) \wedge \tilde{P}_i(x_i) \rightarrow \tilde{P}_i(x_i + 1)) \end{aligned}$$

Otherwise, if Q_i is a universal quantifier

$$\begin{aligned} \eta_i \equiv & (\tilde{P}_{i-1}(x_i, a_{i-1}) \rightarrow \tilde{P}_i(x_i + 1)) \wedge (\tilde{P}_i(x_i) \rightarrow \tilde{P}_i(x_i + 1)) \\ & \wedge (P_{i-1}(x_i, a_{i-1}) \wedge P_i(x_i) \rightarrow P_i(x_i + 1)) \end{aligned}$$

If $Q_i = Q_k$ (it is the outermost quantifier), then $\phi_i = \phi_{i-1} \wedge \alpha_i \wedge \beta_i \wedge \eta_i \wedge \zeta$, where

$$\zeta \equiv P_i(a_i).$$

The resulting formula will be in $SO\exists$-Horn form.

Remark 3.3. In the construction above, it is not, in general, necessary to convert the outermost universal quantifiers, since the corresponding variables can be treated as free variables. We can get a $SO\exists$-Horn formula by converting quantifiers up to the outermost existential quantifier, and then propagating the remaining universal first-order quantifiers to place them in front of the constructed universal first-order quantifiers, so that all second-order quantifiers are in front. In this case, the end condition is stated for the outermost existential quantifier. The example 3.2 illustrates the weak conversion.

We will prove the correctness of the strong conversion. Clearly, the correctness of the strong conversion implies the correctness of the weak: if we have a $SO\exists$-Horn formula and we universally quantify some of its free variables, then, by the corollary 3.2, we can convert it to an equivalent $SO\exists$-Horn formula by moving the universal quantifiers inwards. For the other direction, if our construction is correct, then conversion of the outer universal first-order quantifiers does not change the truth value of the formula.

Example 3.2. Let

$$\psi \equiv \forall x < a \exists y < b \forall z < c\ \phi(x, y, z).$$

Weak conversion produces the following $SO\exists$-Horn formula:

$$\exists P_y \leq a(b+1) \exists \tilde{P}_y \leq a(b+1) \exists P_z \leq ab(c+1) \exists \tilde{P}_z \leq ab(c+1) \forall x < a \forall y < b \forall z < c$$

$$i = 1 \begin{cases} P_z(x,y,0) \wedge (\neg P_z(x,y,0) \vee \neg \tilde{P}_z(x,y,0)) & \alpha_1 \\ \wedge(\neg P_z(x,y,z+1) \vee \neg \tilde{P}_z(x,y,z+1)) & \beta_1 \\ \wedge(\neg\phi(x,y,z) \to \tilde{P}_z(x,y,z+1)) \wedge (\tilde{P}_z(x,y,z) \to \tilde{P}_z(x,y,z+1)) & \eta_1 \\ \qquad \wedge(\phi(x,y,z) \wedge P_z(x,y,z) \to P_z(x,y,z+1)) & \end{cases}$$

$$i = 2 \begin{cases} \wedge \tilde{P}_y(x,0) \wedge (\neg P_y(x,0) \vee \neg \tilde{P}_y(x,0)) & \alpha_2 \\ \wedge(\neg P_y(x,y+1) \vee \neg \tilde{P}_y(x,y+1)) & \beta_2 \\ \wedge(P_z(x,y,c) \to P_y(x,y+1) \wedge (P_y(x,y) \to P_y(x,y+1)) & \eta_2 \\ \qquad \wedge(\tilde{P}_z(x,y,c) \wedge \tilde{P}_y(x,y) \to \tilde{P}_y(x,y+1)) & \\ \wedge P_y(x,b) & \zeta \end{cases}$$

Theorem 3.4. *Let ψ be a Σ_0^B formula with k quantifiers. Let Ψ be the formula obtained from ψ by the above construction. Then*

$$V_1\text{-}Horn \vdash \psi \leftrightarrow \Psi,$$

and Ψ is a $SO\exists$-Horn formula.

Proof. First, we need the following lemmata:

Lemma 3.5. *Let a be a constant, and X and $\tilde{X}$ some sets, such that V_1-Horn $\vdash \forall i < a(X(i) \leftrightarrow \neg\tilde{X}(i)$ We define P using Σ_1^B-Horn-COMP on the conjunction Φ of the following formulae:*

1. $\neg P(0)$

2. $\forall i < a(P(i) \to P(i+1))$,

3. $\forall i < a(X(i) \to P(i+1))$

4. $\forall i < a(P(i+1) \wedge \tilde{X}(i) \to P(i))$.

Then

$$V_1\text{-}Horn \vdash \Phi(P) \to (P(a) \leftrightarrow \exists i < aX(i)).$$

Also, the defining formula for P is $SO\exists$-Horn with respect to X and X'.

Proof. 1. $(\exists i < aX(i)) \to P(a)$

$(\exists i < aX(i)) \to P(i+1)$. We can show by induction that $\forall x < a(P(x) \to P(x+1)) \wedge P(i) \to P(a)$. Σ_1^B-Horn-COMP guarantees the existence of Z, which is a characteristic string of $\phi(z) \equiv P(z+i)$, where $z+i < a$. Now, $P(i) \to Z(0)$ and $\forall x < a(P(x) \to P(x+1)) \to \forall x+i < a(P(x+i) \to P(x+i+1)) \to \forall z+i < a(Z(z) \to Z(z+1))$. By applying induction to Z we get that $Z(b)$ holds, where $b+z = a$. Since $P(a) \leftrightarrow Z(b)$, $P(a)$ holds.

2. $P(a) \to \exists i < aX(i)$

There is at least one element in P, namely a, and so $|P| > 0$. By the Least Number Principle, there exists the minimal element x of P. Since $P(0)$ is false, $x \neq 0$. Then, there exist i s.t. $i+1 = x$. But $P(i+1) \wedge \tilde{X}(i) \to P(i)$, that is, $P(i+1) \to (P(i) \vee X(i))$. Since x is the minimal element of P, $P(i)$ is false. Thus, $X(i)$ holds, and so there exists at least one element (namely, i) in X.

□

Lemma 3.6. *Let P' and $\tilde{P}'$ be two second-order variables, such that on the first a indices $P'(i) \leftrightarrow \neg\tilde{P}'(i)$.*

1. *Suppose P and $\tilde{P}$ satisfy initialization, restriction, transition and propagation rules for converting an existential quantifier, with P' and $\tilde{P}'$ as P_{i-1} and $\tilde{P}_{i-1}$ and bound a. Then $P(a)$ is true iff there is at least one element in P', and $\tilde{P}(a) \leftrightarrow \neg P(a)$*

2. *Suppose P and $\tilde{P}$ satisfy initialization, restriction, transition and propagation rules for converting a universal quantifier, with P' and $\tilde{P}'$ as P_{i-1} and $\tilde{P}_{i-1}$ and bound a. Then $P(a)$ is true iff there is at least one element in $\tilde{P}'$, and $\tilde{P}(a) \leftrightarrow \neg P(a)$*

Proof. Note that in both cases at most one of $P(i)$ and $\tilde{P}(i)$ can hold for every i, by restriction and initialization. If we show that for every $i < a$ one of them holds, they will always have the opposite values. Suppose we are using the rules for an existential quantifier conversion. Then $P(0)$ is false and $\tilde{P}(0)$ is true.

Suppose that at some point both $P(i+1)$ and $\tilde{P}(i+1)$ are false. Σ_1^B-Horn-COMP with $\phi(z) \equiv (\neg P(z) \wedge \neg \tilde{P}(z))$ guarantees the existence of a set of indices $i < a$ on which both P and $\tilde{P}$ are false. Since by assumption this set is nonempty, there is a minimal element x in it by LNP. We know that $x > 0$, since $\tilde{P}(0)$ holds by the initialization. Then, there exists i, such that $i + 1 = x$. Since $P(i+1)$ is false, both $P(i)$ and $P'(i)$ are false by propagation and transition rules. Since $P(i)$ is false and x is the smallest element on which both P and $\tilde{P}$ are false, $\tilde{P}(i)$ holds. Since $\tilde{P}'(i) \wedge \tilde{P}(i) \rightarrow \tilde{P}(i+1)$, $\tilde{P}'(i)$ must be false. But that cannot happen, since by assumption $P'(i) \leftrightarrow \neg\tilde{P}'(i)$. Therefore, $P(x)$ and $\tilde{P}(x)$ always have the opposite values on $x \leq a$, including the case $x = a$.

In order to use lemma 3.5 with P' as X, we need to show its last condition; the other three follow from the transition and propagation rules. Suppose that it does not hold for P, that is, there is an element i for which $P(i+1) \wedge \neg P'(i) \wedge \neg P(i)$. Since $\neg P'(i)$, by assumption $\tilde{P}'(i)$ holds. Since $\neg P(i)$ is true, $\tilde{P}(i)$ holds, because $P(i)$ and $\tilde{P}(i)$ have opposite values. For the same reason, and since $P(i+1)$ holds, $\tilde{P}(i+1)$ should be false, but $\tilde{P}'(i) \wedge \tilde{P}(i) \rightarrow \tilde{P}(i+1)$. Therefore, the last condition of lemma 3.5 holds, and so $P(a)$ holds iff there is an element on which $P'(i)$ is true.

The proof of the second part is almost identical, with P and P' replaced by $\tilde{P}$ and $\tilde{P}'$, and the appropriate rules. $\square$

Now we will show that the following statement holds: if Ψ is obtained from ψ by the strong conversion, then provably $\psi \leftrightarrow \Psi$. Moreover, for $k \geq 1$ the values of the outermost second-order variables P_k and $\tilde{P}_k$ on their bound a_k are such that $P_k(a_k) \leftrightarrow \neg\tilde{P}_k(a_k) \leftrightarrow \Psi$, and there is an end condition clause $P_k(a_k)$. We will prove this statement by induction on the number of quantifiers in ψ. The induction hypothesis will state that we can prove it for the formulae with k quantifiers; the induction step will show that we can prove it then for the formulae with $k+1$ quantifier.

1. When $k = 0$, no conversion is done, so Ψ is identically equivalent to ψ. Moreover, Ψ is a $SO\exists$-Horn formula since it does not have any existentially quantified first-order variables.

2. Suppose $k = 1$. That is, $\psi \equiv Qx < a\phi(x)$, where ϕ is quantifier-free, and all string variables in it are free variables. We need to convert it to

$$\Psi \equiv \exists P \exists \tilde{P} \forall i < a \alpha \wedge \beta \wedge \eta \wedge \zeta.$$

   Σ_1^B-Horn-COMP gives us a string Z corresponding to the values of $\phi(x)$ on $x < a$ (with respect to the variables free in ψ). We can similarly obtain the string $\tilde{Z}$ which represents the complement of Z. Now we can apply the lemma 3.6 with Z and $\tilde{Z}$ corresponding to P' and $\tilde{P}'$. By construction, the values in these P' and $\tilde{P}'$ will be opposite for all indices $< a$. By the lemma 3.6, $P(a) \leftrightarrow \neg\tilde{P}(a)$. If Q was an existential quantifier, $(\exists i < aZ(i)) \leftrightarrow P(a)$. By construction, Z is the string containing witnesses for $\exists x\phi(x)$, so Z is nonempty iff a witness was found. That is, $(\exists i < aZ(i)) \leftrightarrow (\exists i < a\phi(i))$. So, $P(a)$ correctly states the truth of ψ.

   If Q was a universal quantifier, $(\exists i < a\tilde{Z}(i)) \leftrightarrow \tilde{P}(a)$. Since $\tilde{Z}$ contains all x on which $\phi(x)$ is false, $\tilde{Z}$ is nonempty iff a counterexample was found. $\exists i < a\tilde{Z}(i) \leftrightarrow \exists i < a\neg\phi(i) \leftrightarrow \neg\forall i < a\phi(i)$.

3. Assume that for all Σ_0^B formulae with at most k quantifiers the statement above holds, and $k > 1$. We need to show that it holds for a Σ_0^B formula ψ with $k+1$ quantifier. Thus, $\psi \equiv Qx < a\phi(x)$, where $\phi(x)$ is a formula with x as a free variable and k first-order quantifiers. By induction hypothesis, we can convert $\phi(x)$ to obtain $\Phi(x)$, in which the outermost quantifiers P' and $\tilde{P}'$ contain, on their bound b, the value of Φ for various values of x. One of the clauses in Φ is of the form $P'(b)$, corresponding to the end condition. For the purpose of the conversion we remove this clause from Φ, and attach new $\alpha \wedge \beta \wedge \eta \wedge \zeta$. We also introduce existentially quantified SO variables P and $\tilde{P}$, and change Q to $\forall$. Now we need to show that the statement we are proving holds for Ψ. By repeating the discussion for the previous case with P' and $\tilde{P}'$ instead of Z and $\tilde{Z}$ we get that P and $\tilde{P}$ always get the opposite value on $i < a$, and we can apply lemma 3.6 to show that $P(a)$ (and $\tilde{P}(a)$) contain the correct values. That immediately gives us the correctness of the statement.

$\square$

Theorem 3.7. *If ψ is a theorem of V_0, then V_1-Horn $\vdash \psi$.*

Proof. Let a formula ψ be provable in V_0. Since all the axioms and rules of V_0 are contained in V_1-Horn, the proof in V_1-Horn can be the same except for the instances of Σ_0^B-COMP. When such instance is encountered, we can replace a Σ_0^B formula in it by its constructed equivalent (making it an instance of Σ_1^B-Horn-COMP), and attach a proof of their equivalence (from the theorem 3.4). $\square$

Chapter 4

Collapse of V-Horn to V_1-Horn

Intuitively, since $SO\exists$-Horn corresponds to polytime, which is closed under complementation, it should be possible to represent a formula starting with alternating second-order quantifiers by a V_1-Horn formula. We can prove it by formalizing in V_1-Horn the Grädel's proof of the corresponding collapse. The main part of the Grädel's proof is showing that every $SO\exists$-Horn formula with P as a free second-order variable holds for all values of P iff it holds for all P that are false in at most one point. We will not formalize the proof of this claim in V_1-Horn; rather, we will prove the following equivalence directly.

Theorem 4.1. *Let ϕ be a first-order quantifier-free formula. Then.*

$$V_1\text{-}Horn \vdash \forall P \exists Q_k \ldots \exists Q_1 \forall \bar{z} \phi(P, \bar{Q}) \leftrightarrow$$
$$\forall y < t \exists Q_k \ldots \exists Q_1 \forall \bar{z} \phi_y(\bar{Q}) \wedge \exists Q_k \ldots \exists Q_1 \forall \bar{z} \phi_t(\bar{Q}),$$

where ϕ_y is obtained from ϕ by replacing every atom of the form $P(x)$ by an atom $x \neq y$, and in ϕ_t, $P(x)$ is replaced by $x = x$. Also, t is the upper bound on the length of P, and $\bar{z}$ denotes bounded first-order variables.

Proof. The $\rightarrow$ direction is trivial: since $\exists \bar{Q} \forall \bar{z} \phi(P, \bar{Q}, \bar{z})$ holds for any value of P, it should hold for the values of P described by $x \neq y$ and $x = x$. Therefore, all ϕ_y, as well as ϕ_t, will hold.

To prove the other direction, consider the formula $\exists Q_k \ldots \exists Q_1 \forall \bar{z} \phi(P, \bar{Q})$, where P is a free variable. For a fixed P, we can construct the witnesses for ϕ given the corresponding witness for ϕ_y. Each Q_i in ϕ will be a characteristic string of a $SO\exists$-Horn formula $\psi_i(P)$, in which P is a free variable; thus, the existence of it will be given by Σ_1^B-Horn-COMP.

In $\psi_i(P)$, we need to consider two cases: either P holds on all $x < t$, or not. Clearly, in the first case the witnesses for ϕ are witnesses of ϕ_t; in the second case they can be constructed from the witnesses for ϕ_y for different y. Since P can be treated as a free variable, we can check if it holds everywhere using the formula

$$\begin{aligned}\alpha \equiv &\exists S \exists \tilde{S} \forall x < t (S(0)) \wedge (S(t)) \wedge \\ &(\neg P(x) \rightarrow \tilde{S}(x+1)) \wedge (\tilde{S}(x) \rightarrow \tilde{S}(x+1)) \wedge (P(x) \wedge S(x) \rightarrow S(x+1))\end{aligned}$$

We want to be able to refer to the witnesses of ϕ_y for several values of y simultaneously. For that, we move the quantifier $\forall y < t$ inwards past $\exists \bar{Q}$, using lemma 3.1, and rename witnesses of ϕ_t to $\bar{Q}'$ to obtain

$$\exists \bar{Q} \exists \bar{Q}' \forall y < t \phi_y(\bar{Q}^{[y]}) \wedge \phi_t(\bar{Q}').$$

Now, we construct $\psi_i(P, s)$ for each i as

$$\psi_i(P, s) \equiv \forall y < t \alpha \wedge (S(t) \rightarrow Q_i'(s)) \wedge (\tilde{S}(t) \rightarrow \neg P(y)) \wedge (\tilde{S}(t) \rightarrow Q_i^{[y]}(s)).$$

The existence of the characteristic strings Q_i of $\psi_i(P)$ are given by Σ_1^B-Horn-COMP. Now we need to show that they are witnesses for $\exists \bar{Q} \phi(P, \bar{Q})$. As mentioned above, if P holds on all $x < t$, then Q_i' are the desired witnesses; and in that case, by construction, Q_i is equal to Q_i'. To show that Q_i are the correct witnesses for the case where P is falsified somewhere, we need the following lemma.

Lemma 4.2 (based on Grädel's Claim [Grä92]). *Let C be a clause in ϕ. Suppose that P is falsified in at least one point. If C is falsified, then there exist a y, such that $\exists Q_k^y \ldots \exists Q_1^y \phi_y(\bar{Q}^y)$ is false.*

Proof. Since ϕ is a $SO\exists$-Horn formula, there is at most one atom in the conclusion. If C is false, then this atom is false, and all the other atoms in C are true. There are three cases:

1. This atom is of the form $P(x)$ for some x. Then, we take $y = x$
2. This atom is of the form $Q_i(x)$ for some x, i. Then, we take $y \notin P$ such that $x \notin Q_i^y$. Such y exists since P is false in at least one point.
3. Otherwise, we take any $y \notin P$.

Now, we can show that the truth value of C will be the same if all $Q_i(x)$ occurring in C are replaced by $Q_i^y(x)$, and all $P(x)$ are replaced by $x \neq y$. By the choice of y, the conclusion will be false. We know that all atoms in the assumption are true. If $P(x)$ occurs in the assumption, then $P(x)$ holds, and so $x \neq y$. If we replace $P(x)$ by $x \neq y$, the value of C will not change. If $Q_i(x)$ occurs in the assumption, then since $Q_i(x) \to Q_i^y(x)$ for all x and $y \notin P$, and by construction $y \notin P$, $Q_i^y(x)$ holds. Thus, we can replace $Q_i(x)$ by $Q_i^y(x)$ without changing the truth value of C. Any other kind of an atom has a truth value independent from the value of P and Q_i, and so it has the same value in the corresponding clause in ϕ_y. Therefore, C has the same truth value in ϕ_y, and so C is falsified in ϕ_y. Thus, $\exists Q_k^y \ldots \exists Q_1^y \forall \bar{z} \phi_y(\bar{Q}^y)$ is false. $\square$

Now, by the lemma 4.2, if C is a clause of ϕ, then C holds for the given values of the second-order variables. Therefore, all clauses in ϕ hold, and thus ϕ holds. Since P was an arbitrary second-order variable, this shows that $\forall P \exists Q_k \ldots \exists Q_1 \forall \bar{z} \phi$ holds. $\square$

Corollary 4.3. *A formula Ψ with any SO quantifiers and bounded universal FO quantifiers, whose quantifier-free part ϕ is Horn with respect to quantified second-order variables can be provably converted to a $SO\exists$-Horn formula in canonical form.*

Proof. A quantifier-free formula is in $SO\exists$-Horn form. If Ψ has only one quantifier, then it is $SO\exists$-Horn unless the quantifier is SO $\forall$. In this case we replace it by an equivalent

$SO\exists$-Horn formula given by the theorem 4.1, which can be converted to canonical form using corollary 3.2.

Suppose we can provably convert formulae with k quantifiers. Let Ψ be a formula with $k+1$ quantifier and quantifier-free part ϕ which is Horn with respect to quantified SO variables. Consider the subformula under the outermost quantifier. We can treat that variable as a free variable in the subformula, and convert it to obtain a $SO\exists$-Horn formula Φ in canonical form. Note that neither the conversion of the universal SO quantifier nor quantifier exchange introduce new negated variables, so if the outermost quantifier is SO, and Ψ did not have negated occurrences of that variable, Φ will not have any negated occurrences of it either. Therefore, Ψ is equivalent to Φ preceded by the first quantifier of Ψ. Now, that quantifier is FO $\forall$, then we can move it inside using the quantifier exchange rule. If the quantifier is SO $\exists$, we are done, and if it is a SO $\forall$, we can apply theorem 4.1 to obtain a formula that is easily convertible (using quantifier exchange rule) into a $SO\exists$-Horn formula in canonical form. □

Therefore, any formula that does not have FO $\exists$ quantifiers, such that its quantifier-free part is Horn with respect to quantifier second-order variables can be provably converted into a $SO\exists$-Horn formula in canonical form.

Theorem 4.4. *V_1-Horn $\vdash \Sigma^B$-Horn-COMP, where Σ^B-Horn-COMP is a comprehension scheme for SO-Horn formulae.*

Proof. Let Ψ be a SO-Horn formula, used in an instance of Σ^B-Horn-COMP. We can create an equivalent $SO\exists$-Horn formula Ψ', using the proof from corollary 4.3. Then, Σ_1^B-Horn-COMP gives us the existence of the characteristic string of $\Psi \equiv \Psi'$. □

Chapter 5

Encoding a Run of $SO\exists$-Horn-SAT Algorithm

In many cases we need to check a truth value of one SO-Horn formula to determine the value of another. Moreover, often we need to be able to check the values of both the formula itself and its complement. In order to keep our "outer" formula SO-Horn, we have to check them by checking just the values of a constant number of boolean variables. We encountered a similar problem when we converted AC_0 formulae to $SO\exists$-Horn form: in order to convert the outer quantifiers we needed the value of the formula under them. But, in that case, we could negate the quantifier-free part of the formula; we cannot do it for an arbitrary SO-Horn formula.

The technique that will allow us to handle the general case is encoding a run of a $SO\exists$-Horn-SAT algorithm on a given SO-Horn formula by a $SO\exists$-Horn formula, and then checking the result of the run. The encoding will have an additional useful property that all variables in it will occur together with their $\sim$-counterparts.

5.1 $SO\exists$-Horn-SAT Algorithm

Suppose we are given a SO-Horn formula Φ. We want to construct a $SO\exists$-Horn formula Run_Φ which will encode the run of the $SO\exists$-Horn-SAT algorithm on Φ. First of all, because of the SO-Horn hierarchy collapse theorem we only need to consider formulae with $SO\exists$ quantifiers. Moreover, we can represent all $SO\exists$ quantifiers as one $SO\exists$ quantifier corresponding to an array of the original ones. Therefore, we just need to consider the case when

$$\Phi \equiv \exists P \forall x_1 < b_1 \ldots \forall x_l < b_l \phi(P, x_1, \ldots, x_l).$$

Here, ϕ is $SO\exists$-Horn with respect to P. Let m be the number of clauses in ϕ. Let a be the successor of the largest index of P in ϕ, i.e., all indexed elements of P are less than a. In general case, Φ can contain first-order and second-order free variables; to simplify the notation, we will not write them explicitly.

The first step of the $SO\exists$-Horn-SAT algorithm is converting Φ to propositional form by considering $b = b_1 b_2 \ldots b_l$ copies of ϕ, and, in each copy, setting first-order variables to a different l-tuple of values. The conjunction of these copies is equivalent to the original ϕ with first-order quantifiers. That allows us to evaluate all terms and atoms with the first-order relation symbols, i.e., atoms of the form $f(\bar{x}) = g(\bar{x})$, $f(\bar{x}) < g(\bar{x})$, etc. If, at this step, any of the clauses becomes identically equal to $(1 \rightarrow 0)$, then Φ is false, and so the algorithm should return "false". If a clause is valid (i.e., its conclusion becomes identically true, or its assumption identically false), then we do not need to consider this clause. This way, we can remove all atoms with the first-order relation symbols from all clauses, as well as mark off the valid clauses so as not to consider them later.

This results in a conjunction of the clauses that only contain atoms of the form $P(f(v_1, \ldots, v_l))$ for some values $v_1, \ldots, v_l$ of $x_1, \ldots, x_l$. Distinct atoms assume values completely independently, and so can be treated as distinct propositional variables. Now we can run the propositional Horn satisfiability algorithm on the resulting propositional

Horn formula.

We start with all propositional variables $P'(i)$s initialized to 0. On each step of the algorithm we will look for unsatisfied clauses and satisfy them by setting the atoms in their conclusions to 1. This may result in some other clauses becoming unsatisfied. We can view this process as "removing" the atoms we set (and the clauses that become valid under our partial truth assignment) from the formula. A clause is unsatisfied iff its assumption is empty (everything that was there is set). If its conclusion is also empty (as in the case where it contains an atom with a first-order relation which evaluated to 0), then the clause cannot be satisfied and thus the whole formula is unsatisfiable. Otherwise, there is an atom of the form $P(i)$ in the conclusion which can be set to satisfy the clause. Note that we can set atoms in the conclusions of all currently unsatisfiable formulae simultaneously.

We need to repeat these steps until either we find a false clause $(1 \rightarrow 0)$, or there are no unsatisfiable clauses left. Since we need to set at least one atom at each step, the run of the algorithm will take at most a steps. Thus, to check if a satisfying assignment has been found, we only need to check if there is a clause with empty assumption and empty conclusion at step a. If not, the formula Φ evaluated to "true".

5.2 Allowing a Restricted Version of FO$\exists$ Quantifiers

In numerous places in the following construction we need to determine if there exist an index i such that a set of second-order variables $X_1 \dots X_s, Y$ all hold on i. Assuming that "$\sim$"-versions of all these variables are available, we can do it using an extension of the clauses in Lemma 3.5. We will use the following notation:

Assume that we want to find $i < a$ in Y, such that the conditions $X_1(i), \dots, X_s(i)$ hold. For that we will introduce two existentially quantified second-order variables S_Y and $\tilde{S_Y}$ (the search variables are denoted by S and $\tilde{S}$, with the name of the variable in which we are searching as a subscript). We require $S_Y(a)$ to hold iff there is $i < a$, such that both Y and conditions hold on i. As usual, $\tilde{S}_Y$ corresponds to the complement of S_Y. We denote by $\textsc{Search}(a, X_1, \ldots, X_s, Y)$ the conjunction of the following clauses, preceded by $\exists S_Y \exists \tilde{S}_Y \forall i < a$:

1. $\tilde{S}_Y(0)$

2. $(\neg S_Y(i) \vee \neg \tilde{S}_Y(i))$

3. $(X_1(i) \wedge X_2(i) \wedge \ldots \wedge X_s(i) \wedge Y(i) \rightarrow S_Y(i+1)) \wedge (S_Y(i) \rightarrow S_Y(i+1))$

4. $(\tilde{S}_Y(i) \wedge \tilde{X}_1(i) \rightarrow \tilde{S}_Y(i+1)) \wedge \ldots \wedge (\tilde{S}_Y(i) \wedge \tilde{X}_s(i) \rightarrow \tilde{S}_Y(i+1)) \wedge (\tilde{S}_Y(i) \wedge \tilde{Y}(i) \rightarrow \tilde{S}_Y(i+1))$

If we need to use an l-tuple of variables as a search index, we can do so by a simple modification of the above formula: instead of checking all values between 0 and a, we check all l-tuples of possible values. The order does not matter, as long as $(0, \ldots, 0)$ is the first, and $(b_1, \ldots, b_l)$ is the last, where b_i is the bound for the i-th variable.

Remark 5.1. In general, we want to use the variables S_Y and $\tilde{S}_Y$ outside of the search macro itself. For that, the two quantifiers $\exists S_Y \exists \tilde{S}_Y$ need to be moved outside the scope of the formula where S_Y and $\tilde{S}_Y$ are used, possibly applying lemma 3.1, in which case they can acquire additional dimensions.

Thus, S_Y and $\tilde{S}_Y$ would always have a scope larger than that of the search macro. The main reason for placing $\exists S_Y \exists \tilde{S}_Y$ inside the definition of the search macro is simplifying the notation: otherwise, these quantifiers would have to be explicitly mentioned every time the search macro is used, cluttering the formula.

Lemma 5.2. *If S_Y and $\tilde{S}_Y$ are defined in the* $\textsc{Search}(a, \ldots, X_s, Y)$ *macro above, then V_1-Horn proves that $S_Y(a)$ holds (and $\tilde{S}_Y(a)$ does not hold) iff there exists i, such that $Y(i)$ holds and all conditions $X_j(i)$ are satisfied.*

Proof. 1. If there is an index i which satisfies all conditions, then $S_Y(a)$ holds.

Assume $(X_1(i) \wedge \ldots X_s(i) \wedge Y(i)))$ holds. Then, $S_Y(i+1)$ holds. Therefore, S_Y is nonempty. Then, by the Least Number Principle there is a minimal element $i_0 = i+1$ in S_Y. There exist (by Σ_1^B-Horn-COMP) a string Z, such that $\forall j < b(Z(j) \leftrightarrow S_Y(i_0 + j))$, where $b + i_0 = a$. Since $S_Y(i+1)$ holds, $Z(0)$ holds; since $\forall i < aS_Y(i) \rightarrow S_Y(i+1)$, $\forall j < bZ(i) \rightarrow Z(i+1)$. Thus, by induction, $Z(b)$ holds. Since $S_Y(a) \leftrightarrow Z(b)$, $S_Y(a)$ holds.

2. If $S_Y(a)$ holds, all conditions were satisfied for some index i.

Since $S_Y(a)$ holds, S_Y is nonempty. By the Least Number Principle, there is a minimal element i_0 in S_Y. Because of the conditions $(\tilde{S}_Y(0)) \wedge (\neg S_Y(0) \vee \neg \tilde{S}_Y(0))$, $S_Y(0)$ is not set, so $i_0 \neq 0$. Then, there is j, such that $j+1 = i_0$. Consider the behavior of $X_1(j), \ldots, X_s(j), Y(j)$. We assumed that they are not all true for any index, and so they do not all hold on j. Suppose, wlog, that $X_k(j)$ does not hold. But then $\tilde{X}_k(j)$ holds. Since i_0 is the minimal element of S_Y, $S_Y(j)$ does not hold and so $\tilde{S}_Y(j)$ does. Therefore, by the item 4, $\tilde{S}_Y(j+1)$ holds, so by $(\tilde{S}_Y(i_0)) \wedge (\neg S_Y(i_0) \vee \neg \tilde{S}_Y(i_0))$, $S_Y(i_0)$ does not hold, which contradicts our assumption that at least one of $X_1(i_0), \ldots, X_s(i_0), Y(i_0)$ is not satisfied.

□

5.3 Construction

Now we can proceed to construct a formula $\text{RUN}_\Phi(R, \tilde{R})$, encoding the run of the algorithm above. For each second-order variable which we will use, we will create its "$\sim$"-counterpart, so that we can avoid the problems created by inability to negate quantified second-order variables and to use existential first-order quantifiers (we will use SEARCH instead). The table 5.3 lists the variables used in the construction.

Note that R and $\tilde{R}$ are treated as free variables in RUN_Φ. They are intended to be existentially quantified just outside the scope where they are being used. In that sense, they are treated similarly to the search variables in SEARCH. Also, since the value of a formula when the values of all free variables are fixed is either "true" or "false", R and $\tilde{R}$ are boolean variables. To make them set variables, they can be viewed as 1-element (or empty) arrays, so we can think of checking the value of $R(0)$ and $\tilde{R}(0)$ when referring to R and $\tilde{R}$. In most cases we are interested in a set of values of the formula for some range of values of its free variables; then R and $\tilde{R}$, moved outwards using the Quantifier Exchange lemma, become arrays of values of the formula, indexed by these free variables.

First, we need to define the set of clauses for our algorithm. Since we need to consider b copies of ϕ, we can no longer keep the original clauses as clauses, since there will not be a constant number of them. Instead, we will store the information about the atoms in the clauses ($P(i)'$s) in two $l+2$-dimensional matrices, C and D. The matrix C specifies atoms $P(f(\bar{x}))$ appearing in the assumptions of the clauses; the matrix D similarly holds the table of atoms in conclusions. Thus, $C(i, x_1, \ldots, x_l, j)$ is set iff in the assumption of the original clause i there is an atom $P(f(x_1, \ldots, x_l))$ for some f, and $f(x_1, \ldots, x_l) = j$. Our RUN$_\Phi$ will be a conjunction of three parts: initialization (α), step description (β) and end condition ($\gamma \wedge \zeta$). All the formula is preceded by the following quantifiers, for the variables that are used throughout the algorithm:

$$\exists C \exists \tilde{C} \exists D \exists \tilde{D} \exists V \exists \tilde{V} \exists A \exists \tilde{A} \forall t < a$$

Initialization (α). During this step we evaluate first-order relation atoms, define clause matrices C and D, and a validity matrix V. Matrix V will specify which clauses are valid because of the settings of first-order atoms. The initialization part of the formula contains the following quantifiers (with the scope limited to the initialization):

$$\forall x_1 < b_1 \ldots \forall x_l < b_l$$

1. We will place all first-order relation atoms in the assumption, because we can negate them and so we can move such atom from the conclusion to the assumption and negate it. A clause i with first-order atoms $f_{i1}(\bar{x})\Box g_{i1}(\bar{x})\dots f_{ik}(\bar{x})\Box g_{ik}(\bar{x})$, where $\Box$ is one of $=,\neq,<,>,\leq,\geq$, can become valid for a given setting of $\bar{x}$ if at least one atom evaluates to 0. It can become unsatisfiable if there are no other atoms in it, the conclusion is empty and all atoms in the assumptions evaluate to 1. We define $V,\tilde{V}$ as follows:

$$(\neg f_{i1}(\bar{x})\Box g_{i1}(\bar{x}) \rightarrow V(i,\bar{x})) \wedge \dots \wedge (\neg f_{ik}(\bar{x})\Box g_{ik}(\bar{x}) \rightarrow V(i,\bar{x}))$$

$$(f_{i1}(\bar{x})\Box g_{i1}(\bar{x}) \wedge \dots \wedge f_{ik}(\bar{x})\Box g_{ik}(\bar{x}) \rightarrow \tilde{V}(i,\bar{x}))$$

   Note that if there are no atoms of the form $P(i)$ in a clause i, then we can check if it is unsatisfiable for some $\bar{x}$ by checking if $\tilde{V}(i,\bar{x})$ is set for some $\bar{x}$. We can do it with $\textsc{Search}(b_1,\dots,b_l,\tilde{V}(i))$. If there are no first-order atoms in the clause i, we set $\tilde{V}(i)$ for all $\bar{x}$.

2. Now we can define the clause matrices C and D. Let $f_{ij}(\bar{x})$ be the functions such that $P(f_{ij}(\bar{x}))$ occurs in the assumption of i; let $g_i(\bar{x})$ be such that $P(g_i(\bar{x}))$ is in the conclusion of i. We do the following for each clause i:

$$C(i,\bar{x},f_{i1}(\bar{x})) \wedge \dots \wedge C(i,\bar{x},f_{i1}(\bar{x})) \wedge D(i,\bar{x},g_i(\bar{x}))\wedge$$
$$\forall v < a((f_{i1}(\bar{x}) \neq v \wedge \dots \wedge f_{ik}(\bar{x}) \neq v \rightarrow \tilde{C}(i,\bar{x},v)) \wedge (g_i(\bar{x}) \neq v \rightarrow \tilde{D}(i,\bar{x},v)))$$

   Note that if there is no atom $P(v)$ in the assumption of clause i for $\bar{x}$, then instead we can set $C(i,\bar{x})$ with $\forall v < a\tilde{C}(i,\bar{x})$. Similarly, when there is no atom of the form $P(v)$ in the conclusion, we set $\forall v < a\tilde{D}(i,\bar{x})$.

3. We initialize the assignment variable $A(t)$ to $\forall v < a(\tilde{A}(0,v))$

4. We need to force the condition that neither of the variables above can be true

Table 5.1: Variables in the encoding of a run of $SO\exists$-Horn-SAT algorithm

Name	**Size**	**Function**
P	a	The existentially quantified second-order variable in the original formula
C	$m \cdot b_1 \cdot \ldots b_l \cdot a$	$C(i, \bar{x}, v)$ is set iff there is an atom $P(v)$ in the *assumption* of the clause i when first-order variables are set to $\bar{x}$
D	$m \cdot b_1 \cdot \ldots b_l \cdot a$	$D(i, \bar{x}, v)$ is set iff there is an atom $P(v)$ in the *conclusion* of the clause i when first-order variables are set to $\bar{x}$
V	$m \cdot b_1 \cdot \ldots \cdot b_l$	$V(i, \bar{x})$ is set iff the assumption of the clause i evaluates to 0 after setting first-order variables to $\bar{x}$, but prior to setting any of $P(v)$ atoms.
A	$a \cdot a$	$A(t, v)$ is set iff on step t (or earlier) of the execution of the algorithm the propositional variable corresponding to $P(v)$ has been set.
R	1	R is the "result" variable: $\tilde{R}$ is set iff there was an unsatisfied clause.

together with its "$\sim$"-counterpart. Thus, we need to add the following:

$$\begin{aligned}\forall i \leq m \forall x_1 \leq b_1 \ldots \forall x_l \leq b_l (\neg V(i, x_1, \ldots, x_l) \vee \neg \tilde{V}(i, x_1, \ldots, x_l)) \wedge \\ \forall v < a(\neg C(i, x_1, \ldots, x_l, v) \vee \neg \tilde{C}(i, x_1, \ldots, x_l, v)) \wedge \\ (\neg D(i, x_1, \ldots, x_l, v) \vee \neg \tilde{D}(i, x_1, \ldots, x_l, v)) \wedge \\ (\neg A(t, v) \vee \neg \tilde{A}(t, v)) \wedge (\neg R \vee \neg \tilde{R})\end{aligned} \tag{5.1}$$

Step of the algorithm (β). On each step of the algorithm we look for unsatisfied clauses and set the atoms in their conclusions, if we can. A clause may be unsatisfied if it is not valid and all the atoms in the assumption were set on the previous steps. If this condition holds, but the clause is satisfied, then there is an atom in its conclusion that is set. Otherwise, either there is no atom of the form $P(v)$ in the conclusion, and thus we cannot satisfy this clause, or there is such atom there, and it is unset, and so we can satisfy the clause by setting it. Thus, if the clause is not valid and its assumption evaluates to 1, we attempt to find and set the atom in its conclusion.

$$\forall k < a(\tilde{V}(i,\bar{x}) \wedge C(i,\bar{x},v) \wedge D(i,\bar{x},k) \rightarrow A(t+1,k)) \wedge (A(t,k) \rightarrow A(t+1,k)) \quad (5.2)$$

For each value k, we set $\tilde{A}(t+1,k)$ if it was not set at the previous step ($\tilde{A}(t,k)$ holds) and for each clause where it occurs in the conclusion there is at least one unset atom in the assumption. We can check the existence of an unset atom in the assumption of a clause i for $\bar{x}$ with

$$\text{SEARCH}(a, C(i,\bar{x}), \tilde{A}(t)). \quad (5.3)$$

Now, we check that this holds for all clauses by checking, with search macro, if there exists a clause $(i,\bar{x})$, which is not valid ($\tilde{V}(i,\bar{x})$), in the head of which there is an atom $P(k)$ ($D(i,\bar{x},k)$), but there is no unset variable in its assumption ($\tilde{S}_{\tilde{A}(t)}(a)$). Note that $\text{SEARCH}(a, C(i,\bar{x}), \tilde{A}(t))$ is inside the scope of the universal first-order quantifiers $\forall i, \bar{x}$ this statement; thus, the variables $S_{\tilde{A}(t)}$ and $\tilde{S}_{\tilde{A}(t)}$ are local to it. The external search macro is, in turn, under $\forall k < a$. Here, $\sim$ SEARCH(...) means that the condition variable is the negation ("$\sim$"-counterpart) of the search variable of the macro. Now, we specify $\tilde{A}(t+1)$ with

$$\begin{aligned} &\forall k < a\text{SEARCH}(a, b_1, \ldots, b_l, \sim \text{SEARCH}(a, C(i,\bar{x}), \tilde{A}(t)), \bar{V}, D(k)) \wedge \\ &\quad (\tilde{A}(t,k) \wedge \tilde{S}_{D(k)} \rightarrow \tilde{A}(t+1,k)), \end{aligned} \quad (5.4)$$

The resulting description of the step, β, is the conjunction of formulae 5.2 and 5.4.

End condition. There are two possible reasons why Φ can be false. Either one of the clauses of ϕ is identically false under a certain assignment $\bar{x}$ of first-order variables, or no P exists that can satisfy ϕ.

1. The first case occurs when there is a clause with no variables $P(v)$, and in this clause all atoms evaluate to 1. We assume that all such atoms are in the assumption, since if there is a first-order atom in the conclusion, we can negate it and move it to the assumption. There is a constant number of clauses without $P(v)$'s. Such clause is unsatisfiable if $S_{\tilde{V}(i)}$ holds in

$$\textsc{Search}(b_1, \ldots, b_l, \tilde{V}(i)) \tag{5.5}$$

   Note that in this case we are creating a separate pair of search variables $S_{\tilde{V}(i)}$ and $\tilde{S}_{\tilde{V}(i)}$ for each clause i, but since there is a constant number of such clauses, that does not create a problem.

2. In the second case, Φ is false because no P could satisfy ϕ. We can check if the truth assignment satisfies all clauses when the algorithm finishes running, and if it does not, then there is no satisfying truth assignment for the propositional version of Φ and thus P does not exist. In case when such P exists, P is equivalent to $A(a)$.

   The clauses that were satisfied by setting a variable in the conclusion during some step of the execution of the algorithm above will remain satisfied after that step, since all variables in such clauses are set, and $P(v)$ is never forced to change its value from 1 to 0. Thus, there is a step t_0, at which all unsatisfied clauses are unsatisfiable. During t_0 and after it no new variables will be set. So, for all $t > t_0$, the value of A will remain the same, in particular, $A(t_0) = A(a)$. There can be at most a steps during which the value of A changes, since we can set at most a variables. Therefore, to find out if the partial truth assignment at the last step of the execution of the algorithm satisfies all clauses we need to check if $A(a)$ satisfies all clauses.

   The clause is unsatisfiable if all atoms in its assumption are set to 1 and there is no atom $P(v)$ in the conclusion (if there is such $P(v)$, we can set it and satisfy the clause.) Thus, we are searching for a clause that has no unset variables in the assumption (this can be done with the expression 5.3) and an empty conclusion (this can be done with $\text{SEARCH}(a, D(i,\bar{x}))$). The resulting search macro is

$$\gamma \equiv \text{SEARCH}(m, b_1, \ldots, b_l, \tilde{V}, \sim \text{SEARCH}(a, C(i,\bar{x}), \tilde{A}(a)), \sim \text{SEARCH}(a, D(i,\bar{x})))$$

Let us call the resulting search variable S_A. Now, $S_A(m, b_1, \ldots, b_l)$ holds iff some clause is unsatisfiable.

The final end condition ζ specifies the resulting value of Φ by setting R and $\tilde{R}$. If at least one of S_A and $S_{\tilde{V}(i)}$ for the clauses with no $P(i)$'s (assume there are k such clauses) holds, then $\tilde{R}$ is set; otherwise R is set.

$$\begin{aligned}\zeta \equiv &(\tilde{S}_A(m, b_1, \ldots, b_l) \wedge \tilde{S}_{\tilde{V}(i_1)}(b_1, \ldots, b_l) \wedge \ldots \wedge \tilde{S}_{\tilde{V}(i_k)}(b_1, \ldots, b_l) \to R)) \\ &(S_A(m, b_1, \ldots, b_l) \to \tilde{R})(S_{\tilde{V}(i_1)}(b_1, \ldots, b_l) \to \tilde{R}) \ldots (S_{\tilde{V}(i_k)}(b_1, \ldots, b_l) \to \tilde{R})\end{aligned}$$

5.4 Example of the Construction

Example 5.1. Let

$$\begin{aligned}&\Phi \equiv \exists P \forall x_1 < 2 \forall x_2 < 1 \\ &\qquad (x_1 = x_2 + 1 \to P(x_1))(P(x_1 + 1) \to P(x_1 + x_2))(P(x_1 \cdot x_2) \to x_2 < x_1)\end{aligned}$$

This formula is unsatisfiable. We can convert it to a propositional formula with three variables $P(0)$, $P(1)$ and $P(2)$:

$$\begin{aligned}\Phi' \equiv &(0 = 1 \to P(0))(P(1) \to P(0))(P(0) \to 0 < 0) \\ &(1 = 1 \to P(1))(P(2) \to P(1))(P(0) \to 0 < 1)\end{aligned}$$

After we evaluate terms $0 = 1$, $1 = 1$, $0 < 0$ and $0 < 1$, it becomes

$$(P(1) \to P(0))(P(0) \to)(\to P(1))(P(2) \to P(1)),$$

since $(0 = 1 \to P(0))$ and $(P(0) \to 0 < 1)$ are valid clauses. Thus, $V(0,0,0)$ and $V(2,1,0)$ are set.

Now we initialize C and D. In C, we set $C(1,0,0,1)$, $C(1,1,0,2)$, $C(2,0,0,0)$ and $C(2,1,0,0)$. In D, we set $D(0,0,0,0)$, $D(0,1,0,1)$, $D(1,0,0,0)$ and $D(1,1,0,1)$.

In this case, $a = 3$. We initialize $A(0,0) = 0$, $A(0,1) = 0$ and $A(0,2) = 0$. Since there exists a clause (namely, $(1 = 1 \to P(1))$) for which the search condition holds, we set $A(1,1)$. We do not set any other variable. Similarly, at step 1 the clause $(P(1) \to P(0))$ becomes unsatisfied. We set $P(0)$, so $A(2) = (1,1,0)$. Now, at the last step, we would try to satisfy the clause $(P(0) \to)$. This clause cannot be satisfied. So,

$$\textsc{Search}(3,2,1,\tilde{S}_{\textsc{Search}(3,C(i,x_1,x_2),\tilde{A}(3))}(a),\tilde{S}_{\textsc{Search}(3,D(i,x_1,x_2))})$$

makes $S_A(3,2,1)$ true. Thus, $\tilde{R}$ is set in the end condition.

5.5 Correctness Proof

The correctness proof will consist of two parts. First, we will show that, given a $SO\exists$-Horn formula Φ, we can always construct a formula $\textsc{Run}_\Phi$, representing the run of the Horn satisfiability algorithm on Φ. That is, the formula obtained in the construction above always holds. Then, we will show that the result variable R is set iff the formula is satisfiable; otherwise, $\tilde{R}$ is set.

Lemma 5.3. *If* $\textsc{Run}_\Phi(R,\tilde{R})$ *is constructed as above from a SO-Horn formula* Φ*, then*

$$V_1\text{-}Horn \vdash \exists R \exists \tilde{R} \textsc{Run}_\Phi(R,\tilde{R}).$$

Moreover, V_1*-Horn proves that every* $\sim$ *counterpart of a variable evaluates to the complement of that variable.*

Proof. We are trying to show that we can always correctly encode a run of the Horn satisfiability algorithm as a formula. That is, we can always initialize C, D, V and $A(0)$

and their $\sim$-counterparts (α holds), describe a step of an algorithm (β holds), look for an unsatisfied clause (γ holds), and set the corresponding "result" variable (ζ holds).

Consider the initialization part. All of V, C and D are uniquely determined, since an atom is either present in the clause, or absent. We are setting exactly one of $V(i, \bar{x})$ and $\tilde{V}(i, \bar{x})$: first is set iff there is a false first-order atom in the assumption of the clause i for $\bar{x}$, second is set iff all the first-order atoms are true, so at least one is set; the restriction condition 5.1 forces at most one of them to be true. Similarly, $C(i, \bar{x}, v)$ and $D(i, \bar{x}, v)$ are set iff there is an atom $P(v)$ in $i, \bar{x}$, in the assumption or conclusion respectively; otherwise we force their $\sim$-counterparts to hold. Again, condition 5.1 ensures exactly one of them holds. Lastly, since we force all elements of $\tilde{A}(0)$ to hold, $A(0)$ is unset everywhere. Thus, the initialization part uniquely gives us the witnesses for the quantifiers.

Since SEARCH macros work for any conditions, for any values of our variables the clauses composing a SEARCH macro are satisfied. Now we can show that

$$V_1\text{-Horn} \vdash \forall t \leq a \forall v < a(A(t, v) \leftrightarrow \neg\tilde{A}(t, v)).$$

First, $A(t, v)$ and $\tilde{A}(t, v)$ can never be positive at the same time, because of the condition 5.1. Now, suppose there are values of t and v, such that both $\neg A(t, v)$ and $\neg\tilde{A}(t, v)$ hold. Fixing v, we construct a string, containing the values of t for which both variables are false (by Σ_1^B-Horn-COMP), and take the smallest element t_0 of it (by LNP). Now, $t_0 > 0$ since the initialization forces $A(0)$ to hold, so there is an element $t_0' = t_0 + 1$. Since t_0 is the least such element, either $A(t_0', v)$ or $\tilde{A}(t_0', v)$ hold. In the first case, the clause $(A(t, v) \to A(t + 1, v))$ forces $A(t_0, v)$ to hold. In the second case, similarly, either there is a clause with an empty assumption and $P(v)$ in the conclusion, in which case the formula 5.2 sets $A(t_0, v)$; or there is no such clause, then the SEARCH in formula 5.4 sets $\tilde{S}_{D(k)}$, so $\tilde{A}(t_0, v)$ is set. Therefore, at least one of the $A(t, v)$ and $\tilde{A}(t, v)$ is set on each step, so at least one of them is positive; to satisfy the formula 5.1, their values have to be opposite.

The last condition, ζ, sets either R or $\tilde{R}$, but not both. By the same reasoning as in the case of the initialization of V, ζ holds.

Thus, we can always satisfy $\exists R \exists \tilde{R} \textsc{Run}_\Phi(R, \tilde{R})$, and for all second-order variables in it, including R and $\tilde{R}$, their $\sim$ counterparts have the opposite value. $\square$

Lemma 5.4. *Let $\Phi \equiv \exists P \Phi'(P)$ (that is, Φ' denotes the Σ_0^B part of Φ). Then, $A(a)$ provably is the minimal satisfying truth assignment for Φ:*

$$V_1\text{-}Horn \vdash \Phi'(P) \wedge \textsc{Run}_\Phi(R, \tilde{R}) \rightarrow (\forall v < a(A(a,v) \rightarrow P(v))).$$

Proof. Suppose not. Consider all values v, such that $A(a,v)$, but $\neg P(v)$. As given by Σ_1^B-Horn-COMP, we can construct a string X, which will hold these values: $X(v) \leftrightarrow (\neg P(v) \wedge A(a,v))$. Again by Σ_1^B-Horn-COMP, we can construct a string Y which will record steps of the algorithm during which these atoms were set:

$$Y(t) \leftrightarrow (\textsc{Search}(a, \tilde{A}(t), A(t+1), X) \wedge S_X(a)).$$

Here, X is a free variable and so can be negated; its negation will play a role of its $\sim$ counterpart in $\textsc{Search}$, so this formula will be $SO\exists$-Horn. Since we assumed that there are values of $A(a)$ that are not set in P , there is an element in X, and so since this element had to be set on some step of the algorithm by the conditions on A, for some t $\tilde{A}(t,v) \wedge A(t+1,v)$ is satisfied. Therefore, Y is nonempty, and so by the Least Number Principle there is a smallest element t_0 in Y. Now, consider a string Z, which exists by Σ_1^B-Horn-COMP, such that $Z(v) \leftrightarrow \tilde{A}(t_0,v) \wedge A(t_0+1,v) \wedge Y(v)$. Such string contains the atoms in $A(a)$ and not in P that were set first (since if an atom was in $A(t)$ for some t, it propagates to $A(a)$). Now, we take the smallest index v_0 in Z, given by the LNP. Using $\textsc{Search}$, we can find at least one clause in which $P(v_0)$ is in the conclusion, and the assumption evaluates to *true*. Now, since it is the first step on which an index unset in P occurs in $A(t_0+1)$, $\forall v < aA(t_0,v) \rightarrow P(v)$. Therefore, the assumption of this clause evaluates to *true* for P, too. But then, since $P(v_0)$ is in the conclusion, and it is unset, P cannot satisfy this clause. That gives a contradiction, since P is supposed to satisfy all clauses. Thus, if P satisfies the formula ϕ, $\forall v < aA(a,v) \rightarrow P(v)$. $\square$

Theorem 5.5. *Let Φ be a Horn formula, and let R be the result variable from the formula* $\textsc{Run}_\Phi$ *constructed from Φ by the procedure above. Then,*

$$V_1\text{-}Horn \vdash \textsc{Run}_\Phi(R, \tilde{R}) \to (\Phi \leftrightarrow R)$$

Proof.

1. Suppose R is set. By construction, no clause was unsatisfied because of first-order atoms, otherwise one of $S_{\tilde{V}}$ would be set, so $\tilde{R}$ would be set and $(\neg R \vee \neg \tilde{R})$ would not be satisfied. We can show that $P \equiv A(a)$ satisfies ϕ. Suppose not. Then for some values $\bar{x}$ of the first-order variables some clause i is not satisfied by $A(a)$. Then, in row $(i, \bar{x})$ of C for all set variables v, $A(a, v)$ holds, and $D(i, \bar{x})$ is empty. But then $\tilde{S}_A$ does not hold, and so R cannot be set. Therefore, if R is set, then $A(a)$ satisfies ϕ, and Φ holds.

2. Now suppose that R is not set (instead, $\tilde{R}$ is set). Then, either one of $S_{\tilde{V}}$ holds, or S_A holds. In the first case, the formula is unsatisfiable because of first-order atoms, and so no algorithm can find a satisfying assignment. Now suppose S_A holds. That means that there was a clause $(i, \bar{x})$ with the empty (false) conclusion and all atoms in the assumption set. Suppose that there is some satisfying assignment P for ϕ. Since by the lemma 5.4 $A(a) \subseteq P$, P also sets all the atoms in the assumption of clause $i, \bar{x}$. The only situation when the clause can remain unsatisfied, since there are a steps, is when its conclusion is empty. But then P does not satisfy that clause either. So, when the first-order variables evaluate to $\bar{x}$, clause i of ϕ is not satisfied. Therefore, if $A(a)$ is not a satisfying assignment (and, thus, $\tilde{R}$ is set), then ϕ is unsatisfiable and Φ is false.

$\square$

5.6 Complements of $SO\exists$-Horn Formulae

As an example of an application of the construction above, we can define explicitly a complement of a Horn formula. In order to construct $\tilde{\Phi}$, such that $\tilde{\Phi} \leftrightarrow \neg\Phi$, we encode a run of the $SO\exists$-Horn-SAT algorithm on Φ by $\textsc{Run}_\Phi$, and add the condition that $\tilde{R}$ holds, or directly force R and $\tilde{R}$ to be "false" and "true", respectively.

$$\neg\Phi \leftrightarrow \exists R \exists \tilde{R} \textsc{Run}_\Phi(R, \tilde{R}) \wedge \tilde{R} \leftrightarrow \textsc{Run}_\Phi(false, true) \tag{5.6}$$

As shown above, $R \leftrightarrow \Phi$. Since by the definition at least one of R and $\tilde{R}$ is always set, and $(\neg R \vee \neg\tilde{R})$, $R \leftrightarrow \neg\tilde{R}$. Therefore, the formula $\textsc{Run}_\Phi \wedge \tilde{R}$ defines a complement of Φ.

5.7 Allowing Leading First-Order Quantifiers

We can also use this construction to allow first-order quantifiers in front of a $SO\exists$-Horn formula. That is, a formula consisting of a SO-Horn formula preceded by first-order quantifiers is equivalent to a SO-Horn formula. A formula preceded by universal first-order quantifiers is equivalent to a SO-Horn formula by corollaries 3.2 and 4.3. The idea of the construction for the case of existential first-order quantifiers is similar to converting Σ_0^B formulae to $SO\exists$-Horn formulae, but during the innermost step, where we need to check the value and its negation of the quantifier-free part of the formula, we look at the corresponding R and $\tilde{R}$.

So, a $SO\exists$-Horn equivalent of $\exists y < b\Phi(y)$ will be

$$\begin{aligned}\exists Y \exists \tilde{Y} \forall y < b(\neg Y(y) \vee \neg\tilde{Y}(y)) \wedge (\tilde{Y}(0)) \wedge (Y(b)) \wedge \textsc{Run}_\Phi(y) \wedge \\ (R(y,0) \to Y(y+1))(Y(y) \to Y(y+1)) \wedge (\tilde{R}(y,0) \wedge \tilde{Y}(y) \to \tilde{Y}(y+1))\end{aligned}$$

This construction explicitly searches for a witness for $\exists y < b$. A simpler way to represent an existential quantifier is to represent the formula as $\neg\forall y < b\neg\Phi(y)$, then construct a complement of $\Phi(y)$ with $\exists R \exists \tilde{R} \textsc{Run}_\Phi(R, \tilde{R}, y)$. By corollary 3.2, $\forall y < b \exists R \exists \tilde{R} \textsc{Run}_\Phi(R, \tilde{R}, y)$ is equivalent to a SO-Horn formula. So, we can construct a complement of it, using the same method; clearly, the resulting formula will be provably equivalent to $\exists y < b\Phi(y)$.

Chapter 6

Future Work and Conclusions

In this thesis we described an alternative system for feasibly constructive proofs. It would be interesting to formalize some of the known results about systems of the same power, such as QPV, in V_1-Horn, i.e., the relation with Extended Frege proofs. Also, it is known that S_2^1 is finitely axiomatizable, but no such result is known for the systems with the power of QPV; since V_1-Horn does not have infinitely many function symbols, it can be a more suitable model to study this question.

In the rest of this chapter, we give a brief outline of some additional results about V_1-Horn. In particular, we give an intuition for the proof that P-def is equivalent to V_1-Horn.

6.1 Defining Functions in V_1-Horn

Since SO-Horn formulae correspond to polytime functions, we can define functions by stating their defining relations as SO-Horn formulae. Σ^B-Horn-COMP gives the existence of the bit graphs of string functions. Number functions can be treated as string functions with minimization applied to the result.

Using the encoding of the $SO\exists$-Horn-SAT algorithm, we can prove that the functions definable in V_1-Horn are closed under composition, substitution of the result of a function for a variable, and limited recursion. Both for the string and number functions we use the string corresponding to the array of values of the result variable R to check the value of that function (for a particular bit i in the case of string functions).

To show the closure under composition, in the case of number functions we just existentially quantify the value of the first function using the result of section 5.7. In the case of string functions, we construct the new bit graph by similarly checking the bits of the value of the first function. Substitution of a function for a variable is treated similarly. In the case of limited recursion, we use a two-dimensional array to encode the sequence of the result values of the function at each step of the recursion.

Since, as we showed earlier, we can define all AC_0 functions in $SO\exists$-Horn, we obtain a class of functions that includes AC_0 function and is closed under composition and limited recursion; thus, the class of functions we define contains all polytime functions.

6.2 V_1-Horn Is at Least as Powerful as P-def

Using these definitions of functions, we can add function symbols for all polytime functions to V_1-Horn and obtain a conservative extension. Since we proved Σ_0^B-COMP, and the rest of axioms should be easy theorems of V_1-Horn, that will imply that V_1-Horn proves all theorems of P-def.

6.3 V_1-Horn Is at Most as Powerful as P-def

In order to show that P-def is at least as powerful as V_1-Horn, we need to show that P-def can prove every instance of Σ_1^B-Horn-COMP. Since $SO\exists$-Horn-SAT is solvable in polytime, for each $SO\exists$-Horn formula we can find a polytime function that computes the value of this formula. We can then use it to compute the values of this formula on a sequence of indices. The existence of a string containing exactly these values is guaranteed by Σ_0^B-COMP. The resulting string will correspond to the string of values of the original $SO\exists$-Horn formulae, given in V_1-Horn by Σ_1^B-Horn-COMP. So, we can prove in P-def every instance of Σ_1^B-Horn-COMP, and, therefore, all theorems of V_1-Horn.

Therefore, V_1-Horn has the same power as P-def.

Bibliography

[BIS90] D. M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41(3):274 – 30, 1990.

[Bus86] S. Buss. *Bounded Arithmetic*. Bibliopolis, Naples, 1986.

[Bus95] S. Buss. Relating the bounded arithmetic and polynomial time hierarchies. *Annals of Pure and Applied Logic*, 75:67–77, 1995.

[Cob65] A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, editor, *Logic, Methodology and Philosophy of Science*, pages 24–30, Amsterdam, 1965. North-Holland.

[Coo75] S. A. Cook. Feasibly constructive proofs and the propositional calculus. In *Proceedings of the Seventh Annual ACM Symposium on Theory of Computing*, pages 83 –97, 1975.

[Coo98a] S. A. Cook. CSC 2429S: Proof Complexity and Bounded Arithmetic. Course notes, URL: "http://www.cs.toronto.edu/∼sacook/csc2429h", Spring 1998.

[Coo98b] S. A. Cook. Relating the provable collapse of P to NC^1 and the power of logical theories. *DIMACS series in Discrete mathematics and theoretical computer science*, 39:73–91, 1998.

[CU93] S. A. Cook and Alasdair Urquhart. Functional interpretations of feasibly constructive arithmetic. *Annals of Pure and Applied Logic*, 63(2):103 – 200, 1993.

[Fag74] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. *Complexity of computation, SIAM-AMC proceedings*, 7:43–73, 1974.

[Fag93] R. Fagin. Finite-model theory – a personal perspective. *Theoretical Computer Science*, 116:3–31, 1993.

[FSV95] R. Fagin, L. J. Stockmeyer, and M.Y. Vardi. On monadic NP vs. monadic co-NP. *Information and Computation*, 120(1):78–92, 1995.

[Grä91] E. Grädel. The Expressive Power of Second Order Horn Logic. In *Proceedings of 8th Symposium on Theoretical Aspects of Computer Science STACS '91, Hamburg 1991*, volume 480 of *LNCS*, pages 466–477. Springer-Verlag, 1991.

[Grä92] E. Grädel. Capturing Complexity Classes by Fragments of Second Order Logic. *Theoretical Computer Science*, 101:35–57, 1992.

[Imm86] N. Immerman. Relational queries computable in polytime. *Information and Control*, 68:86 –104, 1986.

[KPT91] J. Krajiček, P. Pudlák, and G. Takeuti. Bounded arithmetic and the polynomial time hierarchy. *Annals of Pure and Applied Logic*, 52:143–153, 1991.

[Kra95] J. Krajiček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory.* Cambridge University Press, New York, USA, 1995.

[Par71] R. Parikh. Existence and feasibility of arithmetic. *Journal of Symbolic Logic*, 36:494 –508, 1971.

[PW81a] J. Paris and A.J. Wilkie. Δ_0 sets and induction. In *Proceedings of the Jadwisin Logic Conference*, pages 237–48. Leeds University Press, 1981.

[PW81b] J. Paris and A.J. Wilkie. Models of arithmetic and rudimentary sets. *Bull. Soc. Mathem. Belg., Ser. B*, 33:157 – 69, 1981.

[Raz93] A. Razborov. An equivalence between second-order bounded domain bounded arithmetic and first-order bounded arithmetic. In P. Clote and J. Krajiček, editors, *Arithmetic, proof theory and computational complexity*, pages 247–277. Clarendon Press, Oxford, 1993.

[SP98] U. Schöning and R. Pruim. *Gems of theoretical computer science*. Springer, Berlin, 1998.

[Sto77] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[Tak93] G. Takeuti. RSUV isomorphism. In P. Clote and J. Krajiček, editors, *Arithmetic, proof theory and computational complexity*, pages 364–386. Clarendon Press, Oxford, 1993.

[Var86] M. Vardi. Complexity of relational query languages. *Information and Control*, 68:137 –146, 1986.

[Zam96] D. Zambella. Notes on polynomially bounded arithmetic. *The Journal of Symbolic Logic*, 61(3):942–966, 1996.