Charles University, Prague
Faculty of Mathematics and Physics

Search Problems and Bounded Arithmetic

Jiří Hanika

Ph.D. Thesis

February 2004

Thesis advisor:
Doc. RNDr. Jan Krajíček, DrSc.

Branch M1 — Mathematical Logic

Contents

Acknowledgements

I am deeply grateful to my advisor Jan Krajíček for the variety of mathematical topics he exposed me to, and especially for the vigorous help, advice and every possible support he gave me at all times.

I wish to thank my teachers and colleagues, most notably Pavel Pudlák, Emil Jeřábek, Antonín Sochor, Jiří Sgall, Petr Savický and Vítězslav Švejdar, who shaped my view of relevant mathematics, in various days and ways. The PCMI Summer School in Computational Complexity at the Institute for Advanced Study (Princeton 2000), several Logic Colloquia in recent years and the Workshop in Circuit Complexity and Proof Theory at the International Centre for Mathematical Sciences (Edinburgh 2001), and especially the warm, rich and active research community in Prague also deserve acknowledgement as sources of additional inspiration.

I thank my wife Zuzana for plentiful algebraic, stylistic and other comments which substantially improved this thesis, as well as for her moral support. My thanks go to all my family for encouragement and help.

I acknowledge the continual support of the Institute for Theoretical Computer Science, Prague, (project LN00A056 of MŠMT ČR), of the Mathematical Institute of the Academy of Sciences of the Czech Republic (AV ČR), and of the grant A1019401 of GA AV ČR, as well as some specific travel grants awarded by the US National Science Foundation, the Association of Symbolic Logic, the European Community, the Philosophical Institute of AV ČR, and my alma mater.

Abstract

We study search problems and reducibilities between them with known or potential relevance to bounded arithmetic theories. Our primary objective is to understand the sets of low complexity consequences (esp. Σ^b_1 or Σ^b_2) of theories S^i_2 and T^i_2 for a small i, ideally in a rather strong sense of characterization; or, at least in the standard sense of axiomatization. We also strive for maximum combinatorial simplicity of the characterizations and axiomatizations eventually sufficient to prove conjectured separation results. To this end two techniques based on the Herbrand's theorem are developed. They characterize/axiomatize Σ^b_1-consequences of Σ^b_2-definable search problems, while the method based on the more involved concept of characterization is easier and gives more transparent results. This method yields new proofs of Buss' witnessing theorem and of the relation between PLS and $\Sigma^b_1(T^1_2)$, and also an axiomatization of $\Sigma^b_1(T^2_2)$. We also investigate the relations among known search problems such as GI, MIN and GLS and some of their variants.

Chapter 1

Preliminary Computational Complexity

As computational complexity has become a widely known branch of mathematics, only a few pieces of not unusual terminology and notation will be fixed here. The background for this section can be, for example, [Pap94] or [BDG88], although function problems, our primary object of study, otherwise receive much less than the proportional amount of attention in both these and other introductions to computational complexity theory. We try to amend this deficiency for an interested reader by including a standalone Appendix addressing the relation of function problems to the better understood decision problems.

1.1 Data Representation

We shall work with natural numbers a lot; in fact, most variables and terms throughout the thesis are arithmetic variables and terms. The preferred *encoding* for the purposes of measuring computational complexity resources is *binary*, and so the length of a number is approximately its logarithm[1]. Pairs and other sequences are all encoded in a simple and efficient way, too.

There is perhaps one simple, but essential point to be made here: a

[1]We shall see in Definition 2.1 that the details of the encoding are a little more complicated. The reason is that to receive the full power of both arithmetic and complexity, the correspondence between numbers and bit strings must be a bijection: in particular, prepending a zero bit to a bit string will change the value.

polynomial-time computable function can grow faster than any polynomial. Imagine a number x of length $\log x$ and suppose its representation somehow squares in size (e.g., $\log x$ copies of x are taken). Then its value increases to approximately $x^{\log x}$.

1.2 Computational Model Details

Deterministic Turing machines, usually guaranteed to terminate in time polynomial in the size of the input, and often with an oracle, are used throughout. We always assume that the time bound is enforced by a standard *alarm clock* attached to the machine, making the concrete polynomial syntactically obvious from the machine's description, and the distance from the initial state measured as a number of steps is explicitly present in every recorded *configuration* (i.e., state, head positions and tape contents) of the machine.

A *computation* of a Turing machine is any sequence of adjacent configurations from the initial configuration to any configuration with a terminating state. It is uniquely determined by the machine and the initial configuration, unless the machine invokes an oracle. Note that computations of polynomial-time bounded Turing machines are also polynomial-size objects.

We make no distinction between accepting and rejecting states of a Turing machine when solving search problems[2]; all the search problems under consideration are total and so all terminating states are accepting. There is, however, a dedicated output tape (in addition to the input tape, working tapes and an oracle query tape for every available oracle) whose contents upon entering a terminating state is considered the *output* of a computation.

Oracle *queries* are accessed by writing the query to the oracle tape and shifting to the oracle query state; in this situation the next state is not

[2]In contrast with solving non-total function problems, where rejecting states *must* always be available in order to keep the notion of many-one reducibility compatible with decision problems.

An additional note for an advanced reader. Whenever the *promise* of the search problem is *breached*, the most adequate behavior of the machine solving the problem is indeed a rejection; in particular, if a machine is used as a sub-procedure of another machine, the sub-procedure's rejection forces a rejection by the larger machine by definition. But we adopt a conceptually simpler and computationally slightly *stronger* approach where every answer to an invalid query is valid and there are no rejections.

These approaches are indeed different: some sensitive material to compare them on is Lemma 7.9 and Lemma 7.10.

determined by the transition function, but the tapes and head positions stay unchanged, whereas the machine state shifts to either the *positive* or *negative* state, indicating a positive or negative oracle answer, respectively. If multiple oracles are available, one tape and three states are provided for every oracle. A fully specified oracle can be identified with a *language presented by the oracle*, which is the set of strings for which the oracle indicates the positive answer.

On rare and explicit occasions, where the language presented by the oracle is determined by a formula of the form $\exists y < x\, \varphi(x, y)$ for any oracle query x, the oracle answers will be *witnessed*. That is, not only the state changes as above, but in case of a positive answer, the oracle tape (which just held x) is completely erased and some y is written to its initial segment for which $\varphi(x, y)$ holds. This computational model, however, comes with a price: there may be several computations for a single input, branching by getting different witnesses to positive answers to the same oracle query and so the computed objects are multifunctions[3]. We shall therefore keep such witnesses separate from the (non-witnessed) computations whenever possible.

1.3 Computational Model Notation

Decision problems (i.e., membership in a language) are encountered here especially in connection with oracles. Most prominently, Σ_i^p is the i-th level of the Meyer-Stockmeyer Polynomial Hierarchy; in particular, Σ_0^p is P and Σ_1^p is NP. (In a non-witnessed oracle, the language presented by a Σ_i^p oracle and its complement contribute the same information to the computational model, so we were at a liberty whether to choose Σ_i^p or Π_i^p.)

By $\square_i^p$, $i \geq 1$, the functions computable in deterministic polynomial time with an oracle from Σ_{i-1}^p are denoted; in particular, $\square_1^p$ is FP. Generally, when A is a class of functions defined in terms of some computational model and B is a decision problem (a language), then A^B means the class of functions computed by the same computational model enhanced by an oracle solving B.

When $i \geq 2$, modifications are occasionally employed in which the compu-

[3]In this thesis, both *functions* and *multifunctions* are implicitly total.

An additional note for a curious reader. A *function problem* is actually a partial multifunction: we almost avoid this term, except for the Appendix. A *search problem* is a total function problem, hence a total multifunction.

tational model is enhanced by allowing witnessed queries, but also restricted in the total number of oracle queries. Suppose the input is x and so its length is $n = \log x$; the unrestricted computations can query the oracle at most $n^{O(1)}$ times due to the polynomial time bound. Notation like $\Box_i^p[wit, O(\log n)]$ is then employed to imply $O(\log \log x)$ of witnessed queries.

Chapter 2

Preliminary Bounded Arithmetic

Bounded arithmetic is a field of study centered around specific rather weak formal theories. It started with [Par71]; the area received additional significant potential to strong connections with computational complexity with the Ω_1 axiom introduced in [WP87], which was particularly demonstrated in [Bus86], whose language L_2 became standard. The theory PV was originally ([Coo75]) considered in the context of computational complexity theory and it represents a historically independent thread of research.

Our account of relevant parts of the bounded arithmetic follows [Kra95] with minor differences.

2.1 Basics

We need to present theories whose powers of derivation are the powers of polynomial-time algorithms and to study sometimes very concrete combinatorial structures using them. These structures will always be finite (although we shall also employ infinite classes of such structures to study their asymptotic behavior) and thus the natural numbers suggest themselves as a most natural and most universal host structure for coding everything else. In particular, finite strings of bits continue to be identified with numbers in the binary representation and we use the $|x|$ notation meaning the length of x, i.e., its binary logarithm rounded up in a consistent way, or a convenient primitive symbol $\lfloor \frac{x}{2} \rfloor$ meaning x without the least significant bit.

2.2 Language

The best known axiomatic theories of natural numbers, such as the Peano arithmetic, are formulated in a language centered on addition and multiplication, whose popularity stems directly from the structure of natural numbers. As bounded arithmetic, or rather its specific area linked to computational complexity, was shown by [Bus86] to reflect the power of polynomial-time computable functions, and polynomials are exactly the terms built from addition and multiplication, the reader may expect the same language for bounded arithmetic, too.

However, the polynomial time bounds are not applied to the values of the input, but to the size of the input and so it is polynomial increase of the size of a number and not of its value, which has to be captured by suitable terms in values. Addition in sizes is covered by multiplication in values; multiplication in sizes requires the "smash" in values, the function denoted with # and defined to be $a\#b = 2^{|a|\cdot|b|}$. Furthermore, a few extra functions will simplify the presentation. The language L_2 will thus include: 0, 1, $x+y$, $x \cdot y$, $x\#y$, $\lfloor \frac{x}{2} \rfloor$, $|x|$ and $x \leq y$.

This L_2 now consists of a selected set of polynomial-time computable functions, chosen so that every polynomial-time computable function is majorized by a term in L_2. But it may be sometimes practical also to consider the language L_{PV} which consists of all polynomial-time computable functions, or rather, all their canonical descriptions. If the uninitiated reader decides to skip the following formal definition of L_{PV} at the first reading, she should still read the last paragraph on oracles (relativization) in this subsection.

Definition 2.1 *(Cook [Coo75]) The* language L_{PV} *consists of PV function symbols of any rank $k \in N$. We simultaneously define PV function symbols of rank k, defining equations of rank k, and PV-derivations of rank k, by induction on k. [Text bracketed like this is explanatory, not constitutive.]*

Function symbols of rank 0 *are constant* 0*; unary operators* $s_0(x)$ *[standing for* $2 \cdot x + 1$*],* $s_1(x)$ *[standing for* $2 \cdot x + 2$*], and* $\lfloor \frac{x-1}{2} \rfloor$ *[standing for* $\lfloor \frac{x-1}{2} \rfloor$*], except that* $\lfloor \frac{0-1}{2} \rfloor = 0$*]; and binary operators* $x\#y$ *[standing for a function with approximately the same growth rate as #, namely* $|y|$ *concatenated copies of* x*],* $x \frown y$ *[standing for the concatenation], and* $Less(x, y)$ *[standing for* x *with* $|y|$ *rightmost bits deleted].*

Defining equations of rank 0 *are:*

$$\begin{aligned}
\lfloor \tfrac{0-1}{2} \rfloor &= 0 \\
\lfloor \tfrac{s_0(x)-1}{2} \rfloor &= x \\
\lfloor \tfrac{s_1(x)-1}{2} \rfloor &= x \\
x \frown 0 &= x \\
x \frown s_0(y) &= s_0(x \frown y) \\
x \frown s_1(y) &= s_1(x \frown y) \\
x\#0 &= 0 \\
x\#s_0(y) &= x \frown (x\#y) \\
x\#s_1(y) &= x \frown (x\#y) \\
Less(x,0) &= x \\
Less(x, s_0(y)) &= \lfloor \tfrac{Less(x,y)-1}{2} \rfloor \\
Less(x, s_1(y)) &= \lfloor \tfrac{Less(x,y)-1}{2} \rfloor
\end{aligned}$$

PV rules *are as follows:*

$$\frac{t = u}{u = t}$$

$$\frac{t = u \quad u = v}{t = v}$$

$$\frac{t_1 = u_1, \ldots, t_k = u_k}{f(t_1, \ldots, t_k) = f(u_1, \ldots, u_k)}$$

$$\frac{t = u}{t(x/v) = u(x/v)}$$

$$\frac{\begin{array}{rcl@{\qquad}rcl} f_1(\bar{x}, 0) &=& g(\bar{x}) & f_2(\bar{x}, 0) &=& g(\bar{x}) \\ f_1(\bar{x}, s_0(y)) &=& h_0(\bar{x}, y, f_1(\bar{x}, y)) & f_2(\bar{x}, s_0(y)) &=& h_0(\bar{x}, y, f_2(\bar{x}, y)) \\ f_1(\bar{x}, s_1(y)) &=& h_1(\bar{x}, y, f_1(\bar{x}, y)) & f_2(\bar{x}, s_1(y)) &=& h_1(\bar{x}, y, f_2(\bar{x}, y)) \end{array}}{f_1(\bar{x}, y) = f_2(\bar{x}, y)}$$

PV derivations of rank k *are sequences of equalities in which every function symbol is of rank at most* k *and every of these equalities is either a defining equation of rank at most* k *or derived from some earlier equations by one of the PV rules (i.e., there is a consistent substitution of terms for*

each t,u,v; of PV function symbols for each f, g, h; and variable identifiers for each x, y, so that the considered equation is found under the bar of the rule and all equations found above the bar appear earlier in the sequence).

PV function symbols of rank $k+1$ *are of two kinds, each coming with its respective defining equations. First, for every term t composed of PV function symbols of rank at most k there is a PV function symbol* f_t *of rank* $k+1$. *Second, for every five PV function symbols g,* h_0, h_1, l_0 *and* l_1 *of rank at most k, and every two* π_i*'s* ($i \in \{0,1\}$) *which are PV-derivations of rank at most k of*

$$Less(h_i(\bar{x}, y, z), z \frown l_i(\bar{x}, y)) = 0,$$

there is a PV function symbol $f_{g,h_0,h_1,l_0,l_1,\pi_0,\pi_1}$ *of rank* $k+1$.

PV defining equations of rank $k+1$ *are also of two kinds. For every function symbol* f_t *of the first kind, the defining equation is simply* $f_t = t$. *For every function symbol* $f_{g,h_0,h_1,l_0,l_1,\pi_0,\pi_1}$ *of the second kind, the defining equations are:*

$$f_{g,h_0,h_1,l_0,l_1,\pi_0,\pi_1}(\bar{x}, 0) = g(\bar{x}) \tag{2.1}$$

$$f_{g,h_0,h_1,l_0,l_1,\pi_0,\pi_1}(\bar{x}, s_0(y)) = h_0(\bar{x}, y, f_{g,h_0,h_1,l_0,l_1,\pi_0,\pi_1}(\bar{x}, y)) \tag{2.2}$$

$$f_{g,h_0,h_1,l_0,l_1,\pi_0,\pi_1}(\bar{x}, s_1(y)) = h_1(\bar{x}, y, f_{g,h_0,h_1,l_0,l_1,\pi_0,\pi_1}(\bar{x}, y)). \tag{2.3}$$

Several points should be made here. First, the defining equations of rank 0 are indeed too weak to fix the indicated standard semantics of *PV* function symbols of rank 0, even in conjunction with the rest of the definition. Second, the heart of the definition lies at its end, where closure under *function composition* and *limited recursion on notation* is implemented. Third, the word "limited" in the previous point refers to the condition imposed by the existence of π_0 and π_1; there must be recursion step independent polynomial-time computable limits on the size increase contributed by both h_0 and h_1. For example, h_0 is not allowed to be z^2, as this would make the newly defined function explode to an exponential amount of space; this is enforced by not allowing l_0 or l_1 to depend on z at all. Fourth, every function of rank $k+1$ introduced in this way is uniquely determined, provided its constituent functions of rank at most k are. Fifth, for every *PV* function symbol there are infinitely many other *PV* function symbols denoting a derivably equal function, and additional, non-derivable and both "true" and "false" identities may happen to hold in N, for example even $s_0 = s_1$. This demonstrates why function symbols for each canonical description are

not symbols for each polynomial-time computable function alone. Finally, we get PV function symbols for all functions computable in polynomial time and for no others, as shown already by [Cob65], albeit outside of the formalism of [Coo75].

When the oracle counterparts of the respective theories are being formulated, the language has to be extended with a symbol representing the oracle. It is not essential whether this is a function (as long as its value is a priori bounded by a term in its arguments) or a predicate symbol, whether there are several of them or what their arity is, because all such issues can (and will) be solved with straightforward encoding, preserving relative polynomial time. Thus for any language L_T: $L_T(\alpha)$ is L_T augmented by a single unary predicate symbol α.

2.3 Bounded Quantifier Complexity

To count the bounded quantifier complexity of a formula, first convert it to a prenex form. Three types of quantification are distinguished: unbounded, bounded (by any term) and sharply bounded (by the length of any term). The unbounded ones are forbidden, the bounded ones are counted, the sharply bounded ones are completely ignored. For example, $\exists x < y\ \forall a < |b|\ \exists z < y$ counts as a single block of bounded existential quantifiers.

Definition 2.2 *The* Σ^b_i-formulae *are those which are logically equivalent*[1] *to a prenex formula with no unbounded quantification and only* i *blocks of alternating bounded quantifiers, starting with existential ones. Sharply bounded quantifiers may intervene anywhere, even within the blocks. The* Π^b_i-formulae *are defined in the same way, but starting with universal quantifiers. The* Δ^b_i-formulae over theory T *are those which are* T*-equivalent to both a* Σ^b_i*-formula and a* Π^b_i*-formula. The* Δ^b_i-formulae *without further qualification are the* Δ^b_i*-formulae over* PV_1 *(see below).*

Note that unbounded universal quantifiers are not allowed, but free variables are.

In some places, stricter conditions on sharply bounded quantifier distribution are technically useful.

[1] Equivalent in first order logic. This is a brief way of putting the standard definition at the expense of an apparent deviation from a purely syntactic setting.

Definition 2.3 *A strict*Σ_i^b-formula *is any* Σ_i^b*-formula consisting of* i *alternating bounding quantifiers, the first one being existential, followed by an arbitrary formula in which all quantification is sharply bounded.*

The notation defined in this section is slightly ambiguous as it is applied to formulae in several different languages, even in languages with extraarithmetical symbols. But the language L_2 remains implicit and any other language choice will be clear from the context.

2.4 Theories

The language L_2 is finite and so is the set of basic recursive relations among the symbols as laid out in [Bus86], which will be called $BASIC$:

$$
\begin{aligned}
a + 0 &= a \\
a + (b + c) &= (a + b) + c \\
a \cdot 0 &= 0 \\
a \cdot 1 &= a \\
a \cdot (b + c) &= a \cdot b + a \cdot c \\
a + b &= b + a \\
a \cdot b &= b \cdot a \\
a \leq b \ &\vee \ b \leq a \\
(a \leq b \wedge b \leq a) &\rightarrow a = b \\
(a \leq b \wedge b \leq c) &\rightarrow a \leq c \\
a \leq b &\rightarrow a \leq b + 1 \\
a &\neq a + 1 \\
0 &\leq a \\
a \leq b &\rightarrow a = b \vee a + 1 \leq b \\
2 \cdot a = 0 &\rightarrow a = 0 \\
a &\leq a + b \\
a + b \leq a + c &\rightarrow b \leq c \\
1 \leq a \rightarrow (a \cdot b \leq a \cdot c &\leftrightarrow b \leq c) \\
|0| &= 0 \\
|1| &= 1 \\
a \neq 0 \rightarrow (|2 \cdot a| = |a| + 1 \ &\wedge \ |2 \cdot a + 1| = |a| + 1) \\
a \leq b &\rightarrow |a| \leq |b|
\end{aligned}
$$

$$\begin{aligned}
a \le b \wedge a \neq b &\to 2 \cdot a + 1 \le 2 \cdot b \wedge 2 \cdot a + 1 \neq 2 \cdot b \\
a \neq 0 &\to |a| = ||\lfloor \tfrac{a}{2} \rfloor| + 1 \\
a = \lfloor \tfrac{b}{2} \rfloor &\leftrightarrow 2 \cdot a = b \vee 2 \cdot a + 1 = b \\
0 \# a &= 1 \\
a \neq 0 \to (1\#(2 \cdot a) = 2 \cdot (1\#a) &\wedge 1\#(2 \cdot a + 1) = 2 \cdot (1\#a)) \\
|a\#b| &= |a| \cdot |b| + 1 \\
a\#b &= b\#a \\
|a| = |b| &\to a\#c = b\#c \\
|a| = |b| + |c| &\to a\#d = (b\#d) \cdot (c\#d)
\end{aligned}$$

All of these axioms are open, and the only form of other axioms that appear in theories formulated in L_2 are various types of induction axioms.

Definition 2.4 *The theory T_2^i is axiomatized by the axioms from $BASIC$ and by the induction axioms*

$$(\varphi(0) \wedge \forall y < x\, (\varphi(y) \to \varphi(y+1))) \to \varphi(x),$$

where $\varphi(x)$ is a Σ_i^b-formula in the language L_2.

Definition 2.5 *The theory S_2^i is axiomatized by the axioms from $BASIC$ and by the length induction axioms*

$$(\varphi(0) \wedge \forall y < x\, (\varphi(y) \to \varphi(y+1))) \to \varphi(|x|),$$

where $\varphi(x)$ is a Σ_i^b-formula in the language L_2.

In connection with S_2^i and T_2^i, we always assume $i \ge 1$, even when this is not explicitly stated. The "sharply bounded" theories, which we thus ignore, are much less understood and probably also less useful — due to their stronger dependence on idiosyncratic details of the formulation of $BASIC$.

Definition 2.6 *The union $\bigcup_i T_2^i$ is denoted T_2. Likewise for S_2.*

A principle equivalent to the length induction is the polynomial induction. It is sometimes used instead to define S_2^i:

$$(\varphi(0) \wedge \forall y \le x\, \varphi(\lfloor \tfrac{y}{2} \rfloor) \to \varphi(y)) \to \varphi(x).$$

Another important equivalent formulation replaces the induction axioms by corresponding minimization principles, claiming that a nonempty set definable by the formula has a minimum.

Definition 2.7 *The* Γ-minimization principle *therefore consists of the axiom*

$$\varphi(x) \to \exists y \leq x \, \forall z < y \, (\varphi(y) \wedge \neg\varphi(z))$$

for every $\varphi \in \Gamma$.

Definition 2.8 *The* Γ-length-minimization principle *consists of the axiom*

$$\varphi(x) \to \exists y \leq x \, \forall z < y \, (\varphi(y) \wedge (|z| < |y|) \to \neg\varphi(z))$$

for every $\varphi \in \Gamma$.

Fact 2.9 *(Buss, [Bus86])*

The Σ^b_i*-minimization principle axiomatizes* T^i_2 *over* $BASIC$.

The Σ^b_i*-length-minimization principle axiomatizes* S^i_2 *over* $BASIC$.

Quite a lot is known about these theories. Most notably ([Bus86]),

$$\forall i \;\; S^i_2 \subseteq T^i_2 \subseteq S^{i+1}_2,$$

and therefore $T_2 = S_2$. The tighter inclusion is the second one, in the sense that there is the highest possible conservativity relation short of equivalence. Each S^i_2 or T^i_2 is axiomatized by Σ^b_{i+1}-formulae and S^{i+1}_2 is Σ^b_{i+1}-conservative over T^i_2. So T^i_2 is actually the set of Σ^b_{i+1}-consequences of S^{i+1}_2, as shown by [Bus90].

A weak theory is now introduced over which some of the central concepts will be built. The theory is based on language L_{PV}, i.e., it includes all polynomial-time computable functions. The original PV is a purely equational theory (as indicated in the concept of PV derivations above), but we shall follow [KPT91] in adapting it to the first order logic, while keeping its axiomatics purely universal.

Definition 2.10 *The* theory PV_1 *is axiomatized by all PV-derivable equations in language* L_{PV} *and by the following variant of polynomial induction. For an open formula* $\varphi(x)$ *define a function* $h(b,u)$ *whose values are intervals represented by end points, by induction on length of* u*:*

$$h(b,u) = \begin{cases} (0,b) & \text{if } u = 0 \\ & \text{whenever } u > 0, \text{ assume } h(b, \lfloor \frac{u}{2} \rfloor) = (x,y): \\ (x, x + \lfloor \frac{y}{2} \rfloor) & \text{if } x < x + \lfloor \frac{y}{2} \rfloor \wedge \neg\varphi(x + \lfloor \frac{y}{2} \rfloor) \\ (x + \lfloor \frac{y}{2} \rfloor, y) & \text{if } x + \lfloor \frac{y}{2} \rfloor < y \wedge \varphi(x + \lfloor \frac{y}{2} \rfloor) \\ (x,y) & \text{otherwise} \end{cases}$$

Then the additional axiom of PV_1 *is*

$$(\varphi(0) \wedge \neg\varphi(b) \wedge h(b,b) = (x,y)) \rightarrow (x+1 = y \wedge \varphi(x) \wedge \neg\varphi(y)).$$

This definition is very sketchy and assumes that constructs like $x+y$, $\lfloor \frac{x}{2} \rfloor$, pairing, projection, 1 and definition by cases can be built in L_{PV}. We assume that all these constructs and also all functions from L_2 are identified with some arbitrarily chosen PV function symbols which represent the same functions: for example, 1 with function derived by function composition from $\lfloor \frac{(s_0(s_0(0)))-1}{2} \rfloor$. This is possible as long as special names are given in this way to polynomial-time functions only (cf. [Cob65]).

Everything defined and said so far obviously generalizes to the languages $L_2(\alpha)$ and $L_{PV}(\alpha)$.

2.5 Fractions

Definition 2.11 *Let* Γ *be a class of formulae (for example,* Σ^b_i *for some* i*) and* T *a theory. Then* $\Gamma(T)$ *denotes the consequences of* T *which are also* Γ*.* $\Gamma(T)$ *is called a* fraction *both of* T *and of any* $\Gamma'(T)$ *where* $\Gamma \subseteq \Gamma'$*.*

Note that $\Gamma(T)$ is technically never implicationally closed[2] and that this set of formulae is only interesting for Γ being more restrictive than any known

[2]Unless Γ includes at least all valid sentences of first order logic, no matter how complex.

axiomatization for T: for example, if $\Gamma = \Sigma^b_j$ and $T = S^i_2$, the set of all consequences of $\Sigma^b_{i+1}(S^i_2)$ is equivalent to S^i_2 and the interesting cases of $\Sigma^b_j(S^i_2)$ are those where $j \leq i+1$.

This definition helps to present the already mentioned conservativity result in a terse form as an equivalence of theories:

Fact 2.12 *(Buss [Bus86], [Bus90])*

$$T^i_2 = \Sigma^b_{i+1}(S^{i+1}_2)$$

2.6 Sequence Coding

Although in a weak theory the coding of sequences may be tricky or even impossible, we shall not repeat the standard binary coding of sequences in $I\Delta_0$ developed in [Nel86], [WP87] and [Bus86] and nicely presented in [HP98, Section 5.3][3].

The referenced coding Δ^b_1-defines the operator $(w)_i$ in S^1_2 to extract the i-th element from w if w codes a long enough sequence, and to be 0 otherwise. It also supplies an equally efficient concatenation operator $\frown$ and finite tuple constructor $()$, and lets S^1_2 prove their basic properties, summarized in the following facts.

Fact 2.13 *For a fixed n and i, S^1_2 proves:*

$$(x \frown y) \frown z = x \frown (y \frown z)$$

$$((x_1, \ldots, x_i, \ldots, x_n) \frown y)_i = x_i$$

$$(x_1, \ldots, x_i) \frown (x_{i+1}, \ldots, x_n) = (x_1, \ldots, x_n)$$

Fact 2.14 *Assume $t_1, t_2, \ldots, t_k$ are terms in variables $x_1, x_2, \ldots, x_l$. Then there is a term t^{sup} in the same variables so that $(t_1, t_2, \ldots, t_k) \leq t^{sup}$.*

[3] The last referenced book also includes an easier coding in $I\Delta^{exp}_0$ which is a standard example of an encoding which is *not* usable for our purposes.

Fact 2.15 *Assume $f(x)$, $g(x)$ are Δ^b_1-defined in S^1_2. Then S^1_2 proves that there exists a tuple $(x_1, \ldots, x_{|g(x_1)|})$ such that for any $1 \leq i < |g(x_1)|$,*

$$|f(x_i)| \leq |x_i| + 1 \to x_{i+1} = f(x_i).$$

This tuple is a Δ^b_1-definable function in x_1 and possibly other free variables.

The last fact deserves a bit of comment. Suppose there is an algorithm which computes an n-tuple w with some property in a left-to-right fashion. For the polynomial time bound, measured within the length of its initial configuration x_1, it is enough to show that every element can be obtained from its predecessor by a polynomial-time algorithm f, is larger by at most one bit, and that the total number of such steps n is bounded by a polynomial $n = |g(x_1)|$. This simple proof technique will appear several times, both for constant and non-constant n's.

The principle of *sharply bounded collection* ensures, roughly speaking, that every Σ^b_i-formula can be replaced by an equivalent *strict*Σ^b_i-formula. It is based on sequence coding, too.

Definition 2.16 *The* Γ-sharply-bounded-collection *principle consists of the formula*

$$\forall x < |s|\ \exists z < s'\ \varphi(x, y, z) \iff \exists w < t(y)\ \forall x < |s|\ (w)_x < s' \wedge \varphi(x, y, (w)_x)$$

for every $\varphi \in \Gamma$.

It will be shown available in strong enough theories where needed.

Chapter 3

Search Problems

In contrast with the mainstream of computational complexity theory which concentrates on decision problems (i.e., partial one-valued functions), we study search problems (i.e., total multifunctions with polynomial-size values) in this thesis. Search problems can be mathematically identified with (total) function problems, but the term is normally only used in connection with often simple "problem templates" into which arbitrary procedures from a given class are substituted, and this is what the methods in the area are tailored to. The general relation between decision problems and function problems is far beyond the scope of this thesis, but as we consider several basic points of the traditional treatment of this area in standard computational complexity textbooks traditionally inaccurate, we provide an Appendix addressing these points.

Search problems have been studied by several authors in the past. From the standpoint of pure complexity theory, [JPY88] introduced several classes of search problems including PLS; an important representative of typical combinatorial knowledge in this area is also [BCE+95]. The somewhat more focused interests of researchers in bounded arithmetic obtained a lot of interesting results, most notably [Rii93], [CK98] and [CK99].

3.1 Basic Framework

Definition 3.1 *A* search problem *is a binary relation $R(x, y)$ such that*

1. $\forall x \, \exists y \, R(x, y)$ *and*

2. $\exists k\, \forall x\, \forall y\, (R(x,y) \rightarrow |y| < |x|^k)$

Every x represents an instance *of R and every y such that $R(x,y)$ holds is a* solution *for the instance of R represented by x.*

This definition is the same as the definition of a *total* search problem given in, for example, [BCE+95], but without their general requirement that R be computable in polynomial time.

In our case, R will always be definable in a weak theory of arithmetic, that is, represented by a formula $\varphi(x,y)$ satisfying $\{(x,y) \in N^2; \varphi(x,y)\} = R$, and the conditions above will be provable in the theory (for a fixed k; $|x|^k$ is therefore expressible using k-fold multiplication).

The reader might expect us to give interesting examples of search problems literally conforming to the above definition; if so, she will be disappointed, for the definition will be typically used in a formally different way. Whenever a specific search problem is defined, the description will be parametrized by other predicates or functions taken from the class of polynomial-time computable functions, or sometimes from a higher class $\Box_i^p$. This way the definiendum will be a class of search problems, rather than a single one. It is these highly "uniform" classes of search problems that will be investigated, and the proofs will almost always depend on this uniformity of description.

Definition 3.2 *A class of search problems is* uniformly defined *by a formula φ in an extension of L_2, if it consists exactly of search problems defined by φ into which polynomial-time computable functions and relations are substituted for all the extraarithmetical symbols (i.e., symbols outside L_2).*

It is even safe to speak about the bounded quantifier complexity of φ.

Lemma 3.3 *Let $i \geq 1$. If the bounded quantifier complexity of φ before the substitution of particular polynomial-time functions is Σ_i^b (Π_i^b, Δ_i^b), then suitable definitions can be chosen for each occurrence keeping the resulting formula also Σ_i^b (Π_i^b, Δ_i^b) after the substitution.*

Proof: Polynomial time computability implies definability by a Δ_1^b-formula; so all the substituenda have both Σ_1^b- and Π_1^b-definitions available, to be understood as the *existential* and the *universal* definition, respectively.

Rewrite φ to a logically equivalent prenex formula with propositional connectives limited to negations and conjunctions. If all bounded quantifiers in it are sharply bounded, prepend a dummy quantifier — either $\exists a < b$ or $\forall a < b$ where a nor b occur in φ. Choose the type of the quantifier so as to keep membership in the target class of formulae; in the special case of Δ_i^b where there is no easy choice, perform the following argument independently for Σ_i^b and Π_i^b.

For each occurrence to be substituted for, identify the nearest enclosing quantifier which is not sharply bounded, and the parity of negations between the occurrence and the quantifier. Insert the definition whose type matches that quantifier if the number of negations is even and insert the opposite definition if the number of negations is odd.

In the above-mentioned special case when the target class is Δ_i^b and, before the substitution, φ contained no quantification other than sharply bounded, it is also necessary to verify that the Π_i^b-formula and Σ_i^b-formula obtained from two independent executions of this procedure are equivalent in S_2^1. That is a consequence of the fact that the existential definition and the universal definition of a symbol obtained as above are indeed equivalent in S_2^1, and so are the original Σ_i^b-formula and Π_i^b-formula. $\dashv$

In this context the following convention is introduced:

Definition 3.4 *Suppose A is a uniformly defined class of search problems. Then $A^{\Sigma_i^p}$ is a class defined likewise, but the substituted predicates or functions are $\Box_{i+1}^p$, i.e., computable in polynomial time with an oracle from the indicated level of polynomial hierarchy. As a special case, A^{NP} denotes the class of search problems where predicates and functions which are in P^{NP} and $\Box_2^p$, respectively, are substituted.*

Fact 3.5 *If A is uniformly defined by φ, then every $A^{\Sigma_i^p}$ is uniformly definable by some ψ, whose quantifier complexity exceeds that of φ by at most i.*

Lemma 3.6 *Suppose a search problem is uniformly defined by a bounded formula, so its well-definedness is expressed as $\forall x \exists y \varphi(x, y)$. Then there is a formula equivalent to φ in PV_1 with the same bounded quantifier complexity as φ and with all bounding terms identical (except for the additional outermost length operator of sharply bounded quantifiers), including a bound for the existential quantifier $\exists y$, which asserts the existence of solutions.*

Proof: To see this, first use the Parikh's theorem from [Par71] (also to be found in [HP98] or [Kra95]) to bound the external existential quantifier by a term in x, too. Then proceed from the outside by substituting the bound for each individual variable into all bounds within its scope. For example,

$$\forall x\, \exists y < x^2\, \forall z < x \cdot (y+2)\, \varphi(x,y,z)$$

changes into

$$\forall x\, \exists y < x^2\, \forall z < x \cdot (x^2+2)\, \varphi(x,y,z).$$

This way, no bounding terms refer to variables other than x anymore; but the formula may not be equivalent, as the bounding term may have increased, so the working example has to be polished to

$$\forall x\, \exists y < x^2\, \forall z < x \cdot (x^2+2) \quad z < x \cdot (y+2) \rightarrow \varphi(x,y,z).$$

Dissociate the bounds from the quantifiers and put the sum of all the bounds $D(x)$ in place of each. For every sharp bound $|t(x)|$ contribute $t(x)$ to the sum and treat also the outermost length operator specially to preserve the sharpness. More formally:

$$\begin{array}{rcl}
\forall y < t(x)\, \theta(x,y) & \text{becomes} & \forall y < D(x)\, (y < t(x) \rightarrow \theta(x,y)), \\
\exists y < t(x)\, \theta(x,y) & \text{becomes} & \exists y < D(x)\, (y < t(x) \wedge \theta(x,y)), \\
\forall y < |t(x)|\, \theta(x,y) & \text{becomes} & \forall y < |D(x)|\, (y < |t(x)| \rightarrow \theta(x,y)), \\
\exists y < |t(x)|\, \theta(x,y) & \text{becomes} & \exists y < |D(x)|\, (y < |t(x)| \wedge \theta(x,y)).
\end{array}$$

Equivalence in PV_1 is maintained, because $PV_1 \vdash t(x) \leq D(x)$ and the other necessary inequalities. The quantifier complexity is unchanged. ⊣

By this lemma all the bounding terms may be safely assumed to be equal. This assumption turns out to be natural in important cases and so it is kept throughout the text, starting with the following convention.

Definition 3.7 *The* domain *of a uniformly defined bounded search problem is the set identified by a term into which or into whose length all quantifiers, including the existential quantifier over solutions, are bounded. It is denoted by* $D(x)$.

3.2 Reducibilities

Definition 3.8 *A search problem S is (polynomial-time)* many-one reducible[1] *to a search problem R (denoted as $S \preceq_m R$) if there exist polynomial-time computable functions f and g satisfying*

$$\forall x\, \forall v\, R(f(x), v) \to S(x, g(x, v)).$$

Search problems R and S are (polynomial-time) many-one equivalent[1] *(denoted as $R \approx_m S$) if $R \preceq_m S$ and $S \preceq_m R$.*

It is tempting to claim that decision problems are a special case of search problems with solutions restricted to 0 and 1 and with the property that $\neg(R(x,0) \wedge R(x,1))$. This is not possible in our setting though (cf. Appendix, Myth Three) without breaking the compatibility of many-one reducibility.

Definition 3.9 *(Let* $\operatorname{rng} f^R(x)$ *mean the set of return values of the computations of f with input x for which whenever the oracle answers a y to a query x, $R(x, y)$ holds.) A search problem S is (polynomial-time)* Turing reducible[1] *to a search problem R (denoted as $S \preceq_T R$) if there exists a polynomial-time computable function f with an oracle so that $S(x, y)$ for any $y \in \operatorname{rng} f^R(x)$.*

In the next chapter provable counterparts of some reducibilities over particular arithmetical theories are considered.

3.3 Promise and No Promise

With uniformly defined search problems, it is often natural to split the defining formula $\varphi(x, y)$ informally into the "promise" component $\pi(x)$ and the "search task" component $\varrho(x, y)$. The well-definedness is then expressed as follows:

$$\forall x\, (\pi(x) \to \exists y\, \varrho(x, y)).$$

[1] We shall simply say "many-one reducible", "many-one equivalent", or "Turing reducible", omitting the qualification "polynomial-time". Many-one reductions and Turing reductions in this sense are sometimes called Karp reductions and Cook reductions, respectively.

The promise formula of a search problem can be empty; in that case, the search task alone takes the place of the whole implication. This particularly "pure" situation is termed a "no-promise" search problem. There is a trivial way to derive a (many-one equivalent) no-promise search problem from any search problem, namely to replace the search task by the disjunction of itself and the negation of the promise. The promise tends to be a universal formula and a more radical move is thus available: any initial existential quantification of the negated promise can now be removed and the affected variable renamed (or, affected variables encoded) to y; domain enlargement by a polynomial might be implied. This "overdone no-promise version" no longer preserves many-one equivalence[2], having potential to decrease the complexity of $\varphi(x, y)$, and is *not* what we have in mind when discussing specific search problems. To make this clear we always identify the search task whenever bringing forward a search problem.

Exemplum sequitur.

Definition 3.10 *The search problem MIN is defined by a ternary predicate $<^L (a, x, y)$ conveniently denoted as $x <^L y$, such that for any a, $<^L$ is a linear order and the search task is to find a minimum element of $<^L$.*

Here we have a search problem with a Π_1^b promise (expressing the transitivity, antisymmetry and linearity of $<^L$) and a Σ_2^b search task.

Definition 3.11 *The search problem $\underset{\text{NOPROMISE}}{MIN}$ is again defined by a ternary predicate $x <^L y$. There is no promise and the search task for an a is to find a triple which breaks the transitivity of $<^L$, a pair which breaks antisymmetry or linearity of $<^L$, or an element which is a minimum element of $<^L$.*

This is the "overdone no-promise version" as discussed above. Notice also the details not stated formally: when deriving an instance of a $\underset{\text{NOPROMISE}}{MIN}$ search problem from an instance of MIN, the domain of the search problem is cubed to accommodate the pairs and triples, and additional care is taken in the defining formula that the individual members or pairs effectively stay bounded into the original domain or its Cartesian square.

[2]Unless $P = NP$. Consider the capricious scenario where $\pi(x)$ expresses a $coNP$-complete property of x and $\varrho(x, y)$ is $\pi(x)$.

3.4 Turing versus Many-One

In the cases which primarily interest us it often turns out that the Turing and many-one reducibilities coincide; this is, vaguely speaking, due to the self-compositionality of the problems reduced to. It is however possible to construct pairs of less natural search problems where the many-one reducibility is stronger under reasonable assumptions.

Theorem 3.12 *There are search problems A and B such that $A \preceq_T B$, but $A \preceq_m B \iff P = NP$.*

Proof: The proof just translates the easily seen state of affairs with regard to reducing an FNP-complete function problem (say, graph coloring) to the corresponding NP-complete decision problem (say, graph colorability).

Let A and B be defined by a binary predicate interpreted as an edge in a graph G with the vertex set being encoded in a logarithmic fraction of the domain (say, in unary, as $1, 11_b, 111_b, \ldots, 1^k_b$). We take no care whatsoever about the behavior of the predicate on arguments outside this "graph". Now the search task of B is to "find" an element of the domain which is a vertex of G if and only if G is *not* 3-colorable. The search task of A must satisfy the same, but if G is 3-colorable, the non-vertex n found must also be one which encodes a witness to the 3-colorability (say, there should be a coloring such that the bit of n corresponding to 2^i is 1 if and only if the color of vertex 1^i is black).

Let us see that $A \preceq_T B$. The algorithm will keep asking B about inputs which will be identical to the input to A or almost identical with some new edges added into G. Any such query can be constructed in polynomial time thanks to the size of G.

With the first query to B make sure that G is colorable. If not, we are finished returning the vertex number 1.

If G turned out to be colorable, just keep adding all possible new edges to it one by one; at every step query B to ensure colorability and withdraw the edge if it prevented colorability. After k^2 queries the graph becomes critically colorable: every coloring assigns the same colors to every pair of unconnected vertices. The complement of the graph is therefore a union of three disjoint cliques, each corresponding to a single color and we are finished returning the encoding of the smallest of the cliques, which is always a non-vertex.

Now suppose $A \preceq_m B$ and prove $P = NP$ by extracting a polynomial-time algorithm for 3-coloring from the reduction.

The reduction supposedly works by examining the input, which can encode any graph G (along with some garbage), asking a single query to B (which encodes a graph H) and then effectively constructing a 3-coloring from the oracle answer (whenever there exists one). Consider this algorithm for A: take G. Construct H. Continue the reduction computation twice, once with the assumption that B answered 1 (i.e., denied the 3-colorability of H) and once with the assumption that B answered 0. If one or both branches of the computation assert some particular 3-coloring of G, verify it (that is, check whether no two black vertices are connected and whether the non-black vertices induce a 2-colorable subgraph). If one or both asserted 3-colorings survives this check, output the 3-coloring (which is clearly a correct answer to A no matter whether it was based on an incorrect reply to B — which is quite possible); if both branches deny or fail to witness the 3-colorability, then one of them is based on the right answer to B and thus justifies the negative answer to A. ⊣

Note that the fact that Turing reducibility is allowed to ask multiple queries is not necessarily the key difference from the many-one reducibility: a finite (or even, polylogarithmic) number of queries would be no different from many-one reducibility here. There are however also pairs of search problems, where a trivial Turing reduction with just two queries exists, but a many-one reducibility seems unlikely (compare Lemma 7.9 and Lemma 7.10).

Surprisingly enough search problems seem to be an understudied area of structural complexity and little is known about them along the lines of [LLS75] as yet.

3.5 Relativized Framework

In this section, the concepts presented in other sections of this chapter are extended to their oracle versions; instead of substituting objects for predicate and function symbols into the definition of a uniformly definable class of search problems, we use exactly the unmodified formula (algorithm) and access the objects using an oracle. See also Section 4.3.2 for further discussion and [BCE+95] for an alternative treatment of the same concepts.

Here we often abuse the notation by listing together components of input (parameters) and oracles used to compute an output, as in $R(x, y, \alpha)$. The oracles are distinguished from inputs by being denoted with Greek letters α and β.

The oracle is assumed to be a unary predicate; it is straightforward to accommodate predicates with more arguments, or multiple predicates at the same time using a polynomial-time coding. Function symbols are encoded through binary predicates using the technique of the *bit graph*, lest their logical properties (totality and unique values) are lost. The coding predicate $R_f(x, i)$ is true if and only if the i-th bit of $f(x)$ is 1. It is important that if $R_f(x, i)$ is Δ_1^b-definable, and some predicate S is also Δ_1^b-definable, then there is also a Δ_1^b-definition of S applied to arbitrary terms which may include f. We are going to use this fact implicitly several times.

Definition 3.13 *An* oracle search problem *is a relation $R(x, y, \alpha)$ such that*

1. $\forall\alpha\, \forall x\, \exists y\, R(x, y, \alpha)$ *and*
2. $\exists k\, \forall\alpha\, \forall x\, \forall y\, (R(x, y, \alpha) \rightarrow |y| < |x|^k)$

The concept of definability is similar with the oracle versions; the language of the defining formula now contains α and in place of quantification over these relations we get quantification over models consisting of N enhanced by an extraarithmetical predicate α.

Definition 3.14 *An oracle search problem S is* many-one reducible *to an oracle search problem R (denoted as $S \preceq_m R$) if there exist functions f and g and a relation β computable by a polynomial-time computable function with oracle access to α, satisfying*

$$\forall x\, \forall v\, (R(f(x), v, \beta) \rightarrow S(x, g(x, v), \alpha)).$$

The oracle access to α normally plays an essential role in β, but not in the functions f and g. We do not know whether not publishing the oracle for f and g can make a difference in many-one reducibility with regard to sufficiently robust classes (like polynomial-time computable functions with a specified oracle), but smaller differences *can* occur. For example, the proof of Corollary 8.6 provides a reduction to an identity function using oracle access of f to α. Such a reduction cannot exist with f strictly in polynomial time, but the proof of that corollary would not be affected by such a requirement. The necessary complexity of f could be performed in β instead in that particular situation.

Definition 3.15 *(Let $f^{R(\beta)}(x, \alpha)$ mean the return value of one of the computations of f with input x for which whenever the oracle answers a y to a query x, $R(x, y, \beta)$ holds.) A search problem S is* Turing reducible *to a search problem R (denoted as $S \preceq_T R$) if there exists a polynomial-time computable function f with an oracle and a relation β computable by a polynomial-time function with oracle access to α, so that $S(x, f^{R(\beta)}(x, \alpha))$ for any of the correct computations of f.*

In this definition, f has actually access to two oracles; one represents an oracle input of S, the other one represents an oracle solution of R. This difference of modality is expressed by a difference in notation: the output of f is a solution to information presented through α, not through $R(\beta)$.

Chapter 4

Characterizations

4.1 Many-One Based Characterizations

It is now time to put the search problems and bounded arithmetic together. We shall present several natural concepts of "characterization". Let us begin by the one which has been introduced and studied in [CK98], although with a non-essential modification and with the added ability to refer to its "completeness" part separately, to view *weak characterization* as a special case of a *weak capture* from within.

Definition 4.1 *Let Φ be a class of search problems and T a theory in a language extending L_2. The class* Φ weakly captures *the search problems that are Σ^b_i-definable in T, if and only if for every Σ^b_i-formula $\sigma(x,y)$ for which $T \vdash \forall x\, \exists y\, \sigma(x,y)$ there exists a search problem $S(x,y)$ in Φ and a polynomial-time function g such that*

1. $\forall x\, \exists y\, S(x,y)$
2. $\forall x\, \forall y\, (S(x,y) \rightarrow \sigma(x, g(y)))$

The class Φ weakly characterizes *the search problems that are Σ^b_i-definable in T, if and only if Φ weakly captures them, and for every search problem $R(x,y)$ in Φ there is a Σ^b_i-formula $\varrho(x,y)$ such that*

1. $\forall x\, \forall y\, (\varrho(x,y) \rightarrow R(x,y))$
2. $T \vdash \forall x\, \exists y\, \varrho(x,y)$

Without loss of generality, g can be taken to be a projection function. This is because y can be systematically replaced by $(g(y), y)$ and both this transformation and its inverse can be computed in polynomial time.

Unfortunately this concept, which has been used to derive unprovability results, is not strong enough for the applicability of the Herbrandization methods (Chapter 8 and Chapter 9). In refining it we arrive at the key definition of this thesis:

Definition 4.2 *Let Φ be a class of search problems uniformly defined by some φ and T a theory in a language extending L_2. The class Φ* captures *the search problems that are Σ^b_i-definable in T, if and only if for every Σ^b_i-formula $\varrho(x, y)$, such that $T \vdash \forall x \exists y\, \varrho(x, y)$, there are PV function symbols f, g such that*

$$PV_1 \vdash \forall x\, \forall v\, \varphi(f(x), v) \rightarrow \varrho(x, g(x, v)).$$

The class Φ uniformly defined by φ characterizes *a class Ψ equal to search problems that are Σ^b_i-definable in T, if and only if $\Phi \subseteq \Psi$ and Φ captures all problems in Ψ.*

The basic relation between weak characterization and characterization is that characterization requires the many-one reducibility to be provable in PV_1. This is also the reason why the concept of characterization is restricted to uniformly defined class of search problems[1] in contrast with weak characterization. We shall see that the provability in an open theory is essential for the applicability of the methods developed in Chapter 8 and Chapter 9.

There is also a point in which the weak characterization actually seems stronger than the other one. Recall the weak characterization requirement with regard to an arbitrary formula σ of the right complexity

$$\forall x\, \forall y\, (S(x, y) \rightarrow \sigma(x, g(y)))$$

and its provable counterpart from the concept of characterization:

$$\forall x\, \forall y\, (S(f(x), y) \rightarrow \sigma(x, g(x, y))).$$

It has been already observed that g does not have to depend on x, but the absence of f seems more serious. However, thanks to the absence of a

[1]There is no technical obstacle to extending the notion of characterization to (infinite) unions of uniformly definable sets of search problems with no harm to our results, but making the presentation needlessly awkward.

uniform definability requirement in the definition of weak characterization, it is possible and natural to encode f into S in a way analogous to the way uniform definability works. This way characterization implies weak characterization.

The already known characterization-like results have been obtained using the weak characterization. We shall however need to build on some of them in ways in which the stronger characterization is required: they will be re-proved in this possibly stronger setting in subsequent chapters. The proof methods are to a large extent the same.

4.2 Consequence Based Characterizations

With the characterization definitions of the previous section, a theory T entered the picture only in some kind of a completeness argument concerning a purely combinatorial reduction. A different important approach will now be presented.

Definition 4.3 *A uniformly defined class of search problems S is a* consequence *of a uniformly defined class of search problems R over a theory T, if*

$$T \vdash \forall x\, \exists y\, R(x,y) \rightarrow \forall u\, \exists v\, S(u,v).$$

The theory PV_1 takes the place of T, unless specified otherwise. In the oracle version,

Definition 4.4 *An oracle search problem S is a* consequence *of an oracle search problem R over a theory T, if*

$$\exists \beta\, T \vdash \forall x\, \exists y\, R(x,y,\beta) \rightarrow \forall u\, \exists v\, S(u,v,\alpha)$$

where β is a Δ^b_1-formula in language $L_2 \cup \{\alpha\}$.

The corresponding characterization-like notion is the following:

Definition 4.5 *Let Φ be a union of uniformly definable classes of search problems represented by a set of formulae Φ, and T a theory in language extending L_2. The class* Φ axiomatizes *the search problems that are Σ^b_i-definable in T, if each problem in Φ is Σ^b_i-definable in T, and if*

$$PV_1 + \{\forall x\, \exists y\, R; R \in \Phi\} \vdash \Sigma^b_i(T).$$

The following fact illustrates the relation of the consequence-based characterization to the Turing reducibility, but it will not be used in subsequent proofs and is only included for symmetry with the link between characterization and many-one reducibility.

Fact 4.6 *Let R and S be two classes of search problems, each uniformly definable by a Σ^b_1-formula. If S is a consequence of R, then S is Turing reducible to R.*

Proof: Assume

$$PV_1 \vdash \forall u\, \exists v\, R(u,v) \rightarrow \forall x\, \exists y\, S(x,y).$$

Introduce a new function f which solves R; its computational complexity corresponds to the bounded quantifier complexity of R. Then

$$PV_1 + \forall u\, R(u, f(u)) \vdash \forall x\, \exists y\, S(x,y).$$

By the Buss' witnessing theorem relativized to the language $L_2 \cup \{f\}$ (see Theorem 5.2) there is a polynomial-time machine with oracle access to f solving the search problem S. This machine provides the reduction. $\dashv$

4.3 So What Is the Question?

We have defined two notions of characterization, the many-one reducibility based one and the consequence based one. Each of them can be applied to Σ^b_i-definable search problems in theories S^j_2 or T^j_2 or in their respective oracle versions $S^j_2(\alpha)$ or $T^j_2(\alpha)$. It is even possible to consider Π^b_i-definable search problems instead of the Σ^b_i-definable ones. These alternatives combine into a system of questions: what natural combinatorial problems characterize each class? It is the purpose of this section to sort over all of this.

4.3.1 S_2^j or T_2^j?

Due to the noted Σ^b_{i+1}-conservativity of S_2^{i+1} over T_2^i, these two hierarchies mostly coincide. That is, suppose the Σ^b_i-consequences are targeted and $j \geq i-1$. Then we do not have to distinguish between S_2^{j+1} and T_2^j, as the definitions of the characterization and axiomatization do not distinguish between equivalent fractions. With the consequence characterization, we may even stop distinguishing between fractions $\Sigma^b_i(S_2^j)$ for a fixed j and $i > j+1$; they are all equivalent to S_2^j itself in the sense of axiomatization[2].

This leaves all the interesting difference between S_2^{j+1} and T_2^j to the level of Σ^b_{j+2}-formulae. As this is already above the interesting level (the complexity of the axioms) for T_2^j, it suggests that it is enough to study the slightly more diverse S_2^j hierarchy. We shall however often speak of T_2^{j-1} in place of S_2^j (where the fraction under consideration is equivalent) anyway, sometimes for technical, sometimes for aesthetic reasons.

4.3.2 S_2^j or $S_2^j(\alpha)$?

We are primarily interested in the non-relativized case of S_2^j, which can be obtained from the relativized one by assuming that α is a polynomial-time computable predicate (as are all other members of the language). It turns out that most arguments do relativize and for positive results (which are notoriously more accessible than the negative ones), the non-relativized version is a consequence of the relativized one. Both cases are therefore investigated in parallel to get the stronger result when possible.

While this dichotomy is clear from the logical point of view, some natural computation complexity based approaches can be formulated which apparently lay somewhere in between; for example, instead of oracle access, the additional information could be provided by a Turing machine (of some kind) included in the input.

[2]The (many-one reducibility based) characterization however *does* refer to quantifier complexity of formulae in the fraction. It seems that here the two notions are indeed different from each other as well as they are different from the weak characterization; but we did not pursue this line of research farther, as there is no hope in eventually obtaining any separation results from it.

4.3.3 Many-One or Consequence?

From the computational complexity point of view both reducibilities are of comparable importance, because the consequence reducibility corresponds to the (provable) Turing reducibility at least in the most interesting case of Σ^b_1-consequences.

As the ultimate goal of the field is to prove as much separation between the fractions as there is, the consequence reducibility has immediate merit to be studied. But it again turns out that some positive results can be formulated with the stronger many-one reducibility based notions, and so we did. It is not implied that these two reducibilities generally coincide and some of the remaining open problems may turn out to demonstrate the difference.

The reader may already be aware of the axiomatization of every $\Sigma^b_j(S^i_2)$ and $\Sigma^b_j(T^i_2)$ by the reflection principles for fragments of quantified propositional logic j-RFN(G_i) and j-RFN(G^*_i), respectively, as defined and shown in [KP90]. That is a very different line of investigation. It lacks immediately accessible combinatorial content: the only method of proving separation results which it apparently offers, is thus disproving j-polynomial simulation between quantified propositional calculi. But it does demonstrate in a very elegant way that there is an axiomatization for every fraction under consideration, which is more than it is known concerning the characterizability based on the many-one reduction.

4.3.4 Σ^b_i or Π^b_i?

The definition of a search problem is asymmetric with regard to quantification. The defining formula is placed within existential context, and that may suggest that the Σ^b_i-definable problems in some theory are the same as the Π^b_{i-1}-definable ones: the adjoining existential quantifiers are easily merged in the former case. Well, not exactly: imagine a search problem $\exists z\ \varphi(x, y, z)$ where y is not mentioned in the formula at all, whereas z is something guaranteed to exist in some theory, but hard to find. (See the footnote in Section 3.3 for an example.) If these two existential quantifiers are merged, a different (supposedly harder) Π^b_{i-1} search problem $\varphi(x, (y)_1, (y)_2)$ arises.

On the other hand, almost all characterizations presented in the following chapters avoid using initial existential quantification and thus succeed in characterizing each Σ^b_i-fraction of a chosen theory with a Π^b_{i-1}-definable

problem whenever $i \geq 2$, thus closing the gap between Σ_i^b and Π_{i-1}^b as far as any type of reducibility is concerned. This equivalence does not stretch to the remaining case of Σ_1^b and Π_0^b in the language L_2, because polynomial-time functions have to be substituted for all extraarithmetical symbols, and this requires (some) quantification. Equivalence seems less hopeless for the $i = 1$ case both in the oracle language $L_2(\alpha)$, and in the language L_{PV}; the search tasks of characteristic search problems like PLS can indeed be expressed in a quantifier free way in these cases. But the (negated) promise part of the search problem apparently cannot and so we arrive at the need for Σ_1^b-formulae here even in the extended languages.

Mathematical tradition mostly favors Σ_i^b over Π_i^b in basic investigation and presentation, and so it is followed here. But the attentive reader will notice that Π_i^b-consequences, $strict\Sigma_i^b$-consequences, and $strict\Pi_i^b$-consequences for $i \geq 1$, and Δ_i^b-consequences for $i \geq 2$ are implicitly covered as well.

4.3.5 An Illustrative Summary

We have just reached the boundary between defining our questions and giving answers to them, so it is perhaps worth to summarize them in a table.

<table>
<tr><td>$\vdots$</td><td>$\vdots$</td><td>$\vdots$</td><td>$\vdots$</td><td>$\ddots$</td></tr>
<tr><td>Σ_4^b</td><td colspan="2" rowspan="2"></td><td>?</td><td>$\ldots$</td></tr>
<tr><td>Σ_3^b</td><td>?</td><td>MM</td><td>$\ldots$</td></tr>
<tr><td>Σ_2^b</td><td>$\#FM?$</td><td>FM</td><td>GLS</td><td>$\ldots$</td></tr>
<tr><td>Σ_1^b</td><td>$\square_1^p$</td><td>PLS</td><td>?</td><td>$\ldots$</td></tr>
<tr><td></td><td>S_2^1</td><td>T_2^1 S_2^2</td><td>T_2^2 S_2^3</td><td>$\ldots$</td></tr>
</table>

The table reviews previously known results about the weak characterization as investigated in [CK98]. Each field contains either a search problem to be defined in subsequent chapters, which weakly characterizes the respective class of consequences, or a question mark where no characterization was previously known. The special case of $\#FM$ has a question mark too, because its Σ_2^b-definability in S_2^1 originally relied on an unproven assumption, $\Sigma_2^b(R_3^2) = \Sigma_2^b(S_2^1)$ [3]. An unconditional proof will be given.

[3] Instead of digressing to define the theory R_3^2 here, we refer the interested reader to [Kra95] for additional definitions and to [CK98] for a detailed exposition.

As we study the concept of characterization instead of weak characterization, we start from an empty table like this and try to upgrade the known results to this strongest form wherever possible. The individual characterizations are not obtained independently of each other. It turns out that many results for $\Sigma_j^b(S_2^i)$ are obtained from the respective result for $\Sigma_{j-1}^b(S_2^{i-1})$ and so the table is best understood not in terms of columns or rows, but in terms of lines sloping upwards, especially the diagonal $j = i$, the superdiagonal $j = i + 1$ and the immediate subdiagonal $j = i - 1$; each gets due treatment in one of the following chapters.

Chapter 5

$\Sigma_i^b(S_2^i)$: The Diagonal

5.1 General Background: Buss' Theorem

The key witnessing theorem in bounded arithmetic is the Buss' witnessing theorem. This is partly reflected by the appearances it makes in this text: some related results have already been recalled in the preliminaries, especially the partial conservativity of S_2^{i+1} over T_2^i. The theorem is formulated in this chapter where it is used directly, not for the last time; and new proofs of the theorem shall be given in Section 8.2 and Section 9.2, not to justify it, but to show interesting connections.

The theorem will not be stated here in the maximum strength in which it is known to hold, but only in a version sufficient for our purposes. The interested reader can consult [Bus86] (together with a later improvement in [Bus90]), or [Kra95] for more information on this topic, including the usual proof. On the other hand, we will use the straightforward relativization of the theorem to $L_2(\alpha)$, which is mentioned in the same sources.

Definition 5.1 *Suppose $i \geq 1$ and $\varphi(\bar{x})$ be a Σ_i^b-formula with free variables shown. The predicate $Witness_\varphi^i$ is defined by induction on the logical complexity of φ:*

1. *If $\varphi \in \Sigma_{i-1}^b$ or $\varphi \in \Pi_{i-1}^b$,*

$$Witness_\varphi^i(w, \bar{x}) \iff \varphi(\bar{x})$$

no matter what the following alternatives may suggest.

2. *If* φ *is* $\theta \wedge \chi$,

$$Witness_\varphi^i(w, \bar{x}) \iff Witness_\theta^i((w)_1, \bar{x}) \wedge Witness_\chi^i((w)_2, \bar{x}).$$

3. *If* φ *is* $\theta \vee \chi$,

$$Witness_\varphi^i(w, \bar{x}) \iff Witness_\theta^i((w)_1, \bar{x}) \vee Witness_\chi^i((w)_2, \bar{x}).$$

4. *If* $\varphi(\bar{x})$ *is* $\forall y \leq |t(\bar{x})|\, \theta(\bar{x}, y)$,

$$Witness_\varphi^i(w, \bar{x}) \iff \forall y \leq |t(\bar{x})|\, Witness_\theta^i((w)_{y+1}, \bar{x}, y).$$

5. *If* $\varphi(\bar{x})$ *is* $\exists y \leq t(\bar{x})\, \theta(\bar{x}, y)$,

$$Witness_\varphi^i(w, \bar{x}) \iff (w)_2 \leq t(\bar{x}) \wedge Witness_\theta^i((w)_1, \bar{x}, (w)_2).$$

6. *If* φ *is* $\neg\theta$, *then use double negation elimination and de Morgan and prenex rules to push the odd leading negation into the subformula; then apply one of the preceding clauses.*

This definition transforms $\varphi(\bar{x})$ into another form $\exists w\, Witness_\varphi^i(w, \bar{x})$; the equivalence of these two forms can be shown in reasonably weak theories. The existential quantifier also accumulates a significant amount of total existentiality of φ in the sense that $Witness_\varphi^i(w, \bar{x})$ is a Δ_i^b-formula in S_2^i.

Note that the arity of $Witness_\varphi^i$ depends on the number of free variables in φ, and even that the free variables of $Witness_\varphi^i$ apart from w exactly correspond to those of φ.

Theorem 5.2 *(Buss [Bus86], [Bus90]) Let* $\varphi(x_1, \ldots, x_n)$ *be a* Σ_i^b*-formula with free variables as shown and* $S_2^i \vdash \varphi(x_1, \ldots, x_n)$. *Then there is a function* f *in* $\square_i^p$ *which is* Σ_i^b*-definable in* T_2^{i-1} *and*

$$T_2^{i-1} \vdash Witness_\varphi^i(f(\bar{x}), \bar{x})$$

with PV_1 *instead of* T_2^0 *when* $i = 1$.

The most useful application in connection with search problems is that if φ is $\exists y\, \psi(x, y)$ and satisfies the requirements, solutions are $\square_i^p$ computable: $\varphi(x, (f_\varphi(x))_2)$.

5.2 Function Minimization

We shall give two different characterizations for the diagonal fractions (i.e., $\Sigma_i^b(S_2^i)$) in this section. Each applies to other tastes and they work best for different i's. The first one is perhaps too simple, but it is a search problem anyway.

Definition 5.3 *The search problem* $\Box_i^p$ taken as a search problem *is defined by a formula* $y = f(x)$, *where* $f \in \Box_i^p$. *So there is no promise and the search task is simply to find the value of* f.

Lemma 5.4 *The search problems* Σ_i^b*-definable in* S_2^i *are weakly characterized by the* $\Box_i^p$ *functions taken as search problems.*

Proof: Take any such problem $\varphi(x, y)$. Apply the Buss' theorem to the formula $\exists y\, \varphi(x, y)$, which has a Σ_i^b equivalent by the Parikh's theorem. This yields a function $f(x)$ in $\Box_i^p$ which gives witnesses to $\exists y \varphi(x, y)$. By definition, $\varphi(x, (f(x))_2)$ and so $\varphi(x, y)$ can be reduced to $y = (f(x))_2$ using identity functions. ⊣

Note that for $i = 1$, f is not only in $\Box_i^p$, but the rest of the argument is (also by the Buss' theorem) provable in S_2^1 and so in this particular case we have even the characterization.

Lemma 5.5 *The search problems* Σ_i^b*-definable in* S_2^i *are axiomatized by the* $\Box_i^p$ *functions taken as search problems.*

Proof: By Fact 2.12, it is sufficient to prove the induction for any Σ_{i-1}^b-formula φ in S_2^1 plus the well-definedness of $\Box_i^p$ computations.

Reason in polynomial time with φ as a Σ_{i-1}^b oracle. Suppose $\varphi(0)$ and $\neg\varphi(a)$. Query $\varphi(\lfloor \frac{a}{2} \rfloor)$ etc. using binary search to find x where $\varphi(x)$, but $\neg\varphi(x+1)$. We have arrived at the necessary induction. ⊣

Definition 5.6 *The* function minimization *search problem (FM) is defined by a polynomial-time computable function $f_{FM}(x,y)$ (sometimes conveniently denoted as $f(y)$), so that the search task is to find y where $f_{FM}(x,y)$ is minimal within $D(x)$.*

Some authors have preferred working with the function maximization problem instead. While function maximization and function minimization are trivially many-one equivalent, the function minimization principle fits perhaps more naturally into the family of various search problems more or less based on different kinds of minimization. For this merely didactic purpose we changed the problem and its name, while keeping the same abbreviation.

Lemma 5.7 *Let $i \geq 2$. The $FM^{\Sigma_{i-2}^p}$ search problems are Σ_i^b-definable in S_2^i.*

Proof: The uniformly defining formula of FM problems is Π_1^b; after the substitution of a $\Box_{i-1}^p$ function for f_{FM} (which is represented by a Δ_{i-1}^b-formula) it becomes Π_{i-1}^b.

To prove its well-definedness, consider first the weakening $\psi(x,d)$ *"for some y, $f(y)$ is larger than any other $f(y')$ by at most d"*. As f is polynomial-time computable (albeit with an oracle), its values are bounded by some polynomial $p(x,y)$, too; and this polynomial is majorized by a term $p(x,D(x))$. Clearly, $\psi(x,p(x,D(x)))$ holds and also $\psi(x,d) \rightarrow \psi(x,\lfloor \frac{d}{2} \rfloor)$. Polynomial induction available in S_2^i gives us $\psi(x,0)$, which is the original well-definedness of $FM^{\Sigma_{i-2}^p}$. $\dashv$

Theorem 5.8 *Let $i \geq 2$. The search problems Σ_i^b-definable in S_2^i are weakly characterized by $FM^{\Sigma_{i-2}^p}$ search problems.*

Proof: Lemma 5.7 leaves just the weak capture to be proved.

Take any Σ_i^b-formula $\theta(x,y)$ for which $S_2^i \vdash \forall x\, \exists y\, \theta(x,y)$. By the Buss' theorem, there exists a $\Box_i^p$ function s such that $\theta(x,s(x))$. We need to find an $FM^{\Sigma_{i-2}^p}$ problem $\eta(x,y)$ and a polynomial-time function g satisfying

$$\forall x\, \forall v\, (\eta(x,v) \rightarrow \theta(x,g(x,v))).$$

Take M to be a Turing machine with an oracle from Σ_{i-1}^p which computes $s(x)$ in time $|x|^m$. Define D_x as the set of quintuples of the form:

$$(y,p,w,\epsilon,x)$$

where

1. p is a computation of machine M with input x and output y
2. ϵ is a tuple $\epsilon_1, \ldots, \epsilon_{|x|^m}$ of oracle answers received by the computation p (in order; padded with zeroes at the end)
3. w is a tuple $w_1, \ldots, w_{|x|^m}$ of witnesses to positive oracle answers received by the computation p (so that w_n corresponds to $\epsilon_n = 1$; padded with zeroes elsewhere)

All such tuples have polynomial size (are bounded by a suitable term $D(x)$). The function to be minimized in $D(x)$, say $f(x, c)$ will be defined as $2^{|x|^m}$ if $c \notin D_x$, and as ϵ (encoded as a $|x|^m$-bit number) if $c \in D_x$. The clumsy notation $2^{|x|^m}$ gives actually a number by no more than a single bit longer than any ϵ and trivially computed in linear time: flip all bits of ϵ into zeroes and prepend a one.

The set D_x is always nonempty, because by Fact 2.15 we receive (even in S_2^1) a computation p_0 of M with input x such that all the oracle queries were answered in negative, yielding

$$(y, p_0, (0, \ldots, 0), (0, \ldots, 0), x) \in D_x.$$

Let c be the (y, p, w, ϵ, x) which is $f(x, c)$-minimal in $D(x)$. Obviously all the oracle queries listed in ϵ have been correct: positive answers are all witnessed by the respective w_i's and thus true. If a negative answer $\epsilon_i = 0$ is incorrect and in particular some a would be a witness to the positive answer, then c is not $f(x, c)$-minimal. Use Fact 2.15 to build a computation p' of M with input x and oracle queries answered according to ϵ', which is identical to ϵ, except that whereas $\epsilon_i = 0$, let $\epsilon'_i = 1$. Use a to amend the i'th component of w accordingly, yielding w'. Take y' to be the output of p'. Define

$$c' = (y', p', w', \epsilon', x)$$

and conclude with $c' \in D_x$ and $f(x, c') < f(x, c)$. We have shown that the only minimum of f describes the unique computation of $s(x)$, the witnesses w possibly varying, and its projection to the first coordinate $y = s(x)$ thus satisfies $\theta(x, y)$.

So $\eta(x, y)$ is

$$y \in D_x \wedge \forall y' \in D_x \, f(x, y) \leq f(x, y'),$$

and $g(x, v)$ is $(v)_1$. ⊣

Lemma 5.9 *Let $i \geq 1$ and φ be a Σ_i^b-formula, $s(y)$ and $s'(x,y)$ terms. Then PV_1 together with the statements that $FM^{\Sigma_{i-1}^p}$ is well-defined proves, for some term t, the principle of sharply bounded collection:*

$$\forall x < |s|\ \exists z < s'\ \varphi(x,y,z) \iff \exists w < t(y)\ \forall x < |s|\ (w)_x < s' \wedge \varphi(x,y,(w)_x).$$

Proof: By the same reasoning as in Lemma 3.6, no generality is lost in assuming $s' = s$.

Assume $\varphi \in \Pi_{i-1}^b$ and $\forall x < |s|\ \exists z < s(y)\ \varphi(s(y),x,z)$. Consider the following valuation function:

$$f(y,w) = \begin{cases} |s(y)| & \text{always, if } \exists i < |s(y)|\ (w)_i \geq s(y) \\ |s(y)| - i & \text{if } \forall j < i \leq |s(y)|\ \varphi(y,j,(w)_j) \text{ and not } \varphi(y,i,(w)_i) \\ 0 & \text{if } \forall i < |s(y)|\ \varphi(y,i,(w)_i). \end{cases}$$

By the well-definedness of this particular instance y of the $FM^{\Sigma_{i-1}^p}$ search problem defined by f, there is a w which gets the minimum value of f. Assume the value is positive, some $|s(y)| - i$. Then w provides the first i witnesses by the definition of f. By assumption, $\exists z < s(y)\ \varphi(y,i,z)$. Define

$$(w')_j = \begin{cases} (w)_j & \text{for } j < i, \\ z & \text{for } j = i, \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, $f(y,w') < f(y,w)$, contrary to the minimality; the minimum value is therefore zero, and such w has the desired property — provided t is chosen high enough, using Fact 2.14.

To prove the same for any Σ_i^b-formula, iterate this construction. In each step, the innermost sharply bounded universal quantifier appearing in the initial block of bounded existential quantifiers is pushed to the end of this block.

The converse implication is immediate, provided an efficient coding of sequences is used. $\dashv$

Theorem 5.10 *Let $i \geq 2$. The search problems Σ^b_i-definable in S^i_2 are axiomatized by $FM^{\Sigma^p_{i-2}}$ search problems over theory PV_1.*

Proof: By Fact 2.12, the desired fraction is axiomatized by induction axioms for every Σ^b_{i-1}-formula. Further, it is also axiomatized (see Fact 2.9) by the Σ^b_{i-1}-minimization principles, namely that every nonempty set defined by a Σ^b_{i-1}-formula $\varphi(x)$ has a minimum:

$$\varphi(x) \rightarrow \exists x' \leq x \; \forall y < x' \;\; (\varphi(x') \wedge \neg\varphi(y)).$$

For every such statement build an $FM^{\Sigma^p_{i-2}}$ search problem whose minima immediately encode the (unique) minimum x of the statement.

Take any $\varphi \in \Sigma^b_{i-1}$. Using Lemma 3.6, assume that all the bounding terms depend solely on a single free variable, say x; straightforward formula manipulation also allows us even to assume that φ begins by a block of existential quantification interspersed with sharply bounded universal quantification and continues with a Π^b_{i-2}-subformula.

Force all sharply bounded universal quantification deeper into the formula using Lemma 5.9, finally arriving at some

$$\exists y_1 < t_{y_1}(x) \; \exists y_2 < t_{y_2}(x) \;\; \ldots \exists y_k < t_{y_k}(x) \; \theta(x, y_1, \ldots, y_k).$$

Take a big enough term $t^{sup}(x)$ (see Fact 2.14) and rewrite the formula to

$$\exists y < t^{sup}(x) \; \Big(\bigwedge_i (y)_i < t_{y_i}(x)\Big) \rightarrow \theta(x, (y)_1, (y)_2, \ldots, (y)_k).$$

Ignoring the leading existential quantifier, denote the Π^b_{i-2}-formula so obtained as $\varphi_0(x, y)$.

Let the domain of the desired $FM^{\Sigma^p_{i-2}}$ problem be a set of pairs (x', y) where $x' \leq x$ and $y < t^{sup}(x')$. Define

$$f_{FM}((x', y)) = \begin{cases} x' & \text{if } \varphi_0(x', y), \\ t^{sup}(x) & \text{otherwise.} \end{cases}$$

Recall the definition of Σ^b_{i-1}-minimization principles and assume $\varphi(x)$. Any minimum m found for this function minimization problem satisfies $\varphi(m)$, which is

$$\exists y < t^{sup}(x) \; \varphi_0(m, y),$$

and no $m' < m$ does. This implies the minimality also for $\varphi(m)$, as the *converse* of the described formula manipulation can be performed in PV_1. $\dashv$

Theorem 5.11 *The search problems Σ_1^b-definable in S_2^2 are captured by FM search problems.*

Proof: By Theorem 5.10, any Σ_1^b-definable search problem φ is provable from the well-definedness of FM and PV_1. By the deduction theorem, a finite set of FM search problems exists such that PV_1 plus their well-definedness implies the well-definedness of φ.

Take two such problems, one minimizing function $f_1(x,y)$ in domain $t_1(x)$, the other one minimizing function $f_2(x,y)$ in domain $t_2(x)$. Make the latter's polynomial running time bound explicit as $t(x)$, and define a new FM search problem: its domain is the set of all pairs (x,y) such that $x < t_1(x)$ and $y < t_2(y)$ and its function to be minimized is $f((x,y)) = t(x) \cdot f_1(x) + f_2(x)$. Clearly this construction yields an FM problem, too, and its well-definedness implies the well-definedness of both constituent FM problems. Iterating this construction finitely many times, a finite number of FM problems is characterized by a single one, say, $\forall z < D(x)\ \theta(x,y,z)$ with θ open in L_{PV}. Thus

$$PV_1 \vdash \forall x\ \exists y < D(x)\ \forall z < D(x)\ \theta \rightarrow \forall u\ \exists v\ \varphi(u,v),$$

or, taking u as a constant,

$$PV_1 \vdash \exists x\ \forall y < D(x)\ \exists z < D(x)\ f(x,z) < f(x,y) \vee \exists v\ \varphi(u,v).$$

After applying the Herbrand theorem from [KPT91] to x, z and v, and reintroducing existential quantification over z, we end up with terms t_i and s, such that

$$\begin{aligned} PV_1 \vdash \forall y_1 < D(t_1(u))\ , \ldots , \forall y_k < D(t_k(u, y_1, \ldots, y_{k-1})) & \\ \exists z_1 < t_1(u)\ , \ldots , \exists z_k < t_k(u, y_1, \ldots, y_{k-1}) & \\ \big(f(t_1(u), z_1) \ < \ f(t_1(u), y_1) \quad \vee & \\ f(t_2(u, y_1), z_2) \ < \ f(t_2(u, y_1), y_2) \quad \vee & \\ \ldots \ < \ \ldots \quad \vee & \\ f(t_k(u, y_1, \ldots, y_{k-1}), z_k) \ < \ f(t_k(u, y_1, \ldots, y_{k-1}), y_k) & \\ \vee \ \varphi(u, s(u, y_1, \ldots, y_k))\big), & \end{aligned}$$

or, returning to the original form:

$$PV_1 \vdash \forall y_1 < D(t_1(u)) \,, \ldots, \forall y_k < D(t_k(u, y_1, \ldots, y_{k-1}))$$
$$\Big(\qquad \forall z_1 < t_1(u) \,, \ldots, \forall z_k < t_k(u, y_1, \ldots, y_{k-1})$$
$$\begin{array}{rcll} \Big(\quad f(t_1(u), y_1) & \leq & f(t_1(u), z_1) & \wedge \\ (*) \qquad f(t_2(u, y_1), y_2) & \leq & f(t_2(u, y_1), z_2) & \wedge \\ \ldots & \leq & \ldots & \wedge \\ f(t_k(u, y_1, \ldots, y_{k-1}), y_k) & \leq & f(t_k(u, y_1, \ldots, y_{k-1}), z_k) & \Big) \\ & \rightarrow & \varphi(u, s(x, y_1, \ldots, y_k)) & \Big). \end{array}$$

Now apply the method of Lemma 3.6 to obtain a single term $M(u)$ majorizing all these bounding terms provably in PV_1. Use it to define a new function F as

$$F(u, w) = \sum_{i=1}^{k} f(t_i(u, (w)_1, \ldots, (w)_{i-1}), (w)_i) \cdot M^{k-i}(u),$$

and observe, still inside PV_1, that if the minimum w is some $(y_1, \ldots, y_k)$, then the conjunctive premise of $(*)$ is satisfied. By transitivity of implication, we have proved in PV_1 the many-one reduction (using M^k and s) of φ to the FM search problem specified by F in place of f_{FM}. $\dashv$

Chapter 6

$\Sigma^b_{i+1}(S^i_2)$: The Superdiagonal

6.1 General Background: Bounded Queries

The following result is from [Kra93], also to be found in [Kra95].

Theorem 6.1 *Let $i \geq 1$ and let $\varphi(x, y)$ be a Σ^b_{i+1}-formula such that*

$$S^i_2 \vdash \forall x \, \exists y \, \varphi(x, y).$$

Then there is function $f(x)$ from $\square^p_{i+1}[wit, O(\log n)]$ which is Σ^b_{i+1}-definable in S^i_2 such that

$$S^i_2 \vdash \forall x \, \varphi(x, f(x)).$$

6.2 Sharp Function Minimization

Definition 6.2 *The* sharp function minimization *search problem ($\#FM$) is defined by a polynomial-time computable function $f(x, y)$, so that the search task is to find y such that $|f(x, y)|$ is minimal within $D(x)$.*

Again we have formally deviated from the definition in [CK98], where the analogous *sharply bounded function maximization* effected the exponential shrinking of the range of f by giving a promise that the values of f are bounded by the length of a term. That is because we want to capture more directly the idea that the only difference between FM and $\#FM$ is that the values of f are encoded in binary for FM and in unary for $\#FM$.

Lemma 6.3 *Let $i \geq 2$. The $\#FM^{\Sigma^p_{i-2}}$ search problems are Σ^b_i-definable in S^{i-1}_2.*

Proof: Let $\varphi(x,u)$ be the Σ^b_{i-1}-formula

$$\exists y < D(x)\, |f(x,y)| + 1 > |u|,$$

where $D(x)$ and f are taken from an instance of a $\#FM$ search problem, and let B be the upper bound on the output size derived from f.

Obviously $\varphi(x,0)$ and $\neg\varphi(x,B)$. Therefore

$$\varphi(x, \lfloor \tfrac{z}{2} \rfloor) \wedge \neg\varphi(x,z)$$

for some z. The left hand part of the conjunction provides a y such that

$$|f(x,y)| + 1 > |\lfloor \tfrac{z}{2} \rfloor|$$

and the right hand part of the conjunction guarantees the maximality of this particular value of $|f(x,y)| = |\lfloor \tfrac{z}{2} \rfloor|$. ⊣

Theorem 6.4 *Let $i \geq 2$. The search problems Σ^b_i-definable in S^{i-1}_2 are weakly characterized by $\#FM^{\Sigma^p_{i-2}}$ search problems.*

Proof: Lemma 6.3 leaves just the weak capture to be proved.

The proof is a variant of the proof of Theorem 5.8. Instead of Buss' theorem we start from Theorem 6.1 and so we find ourselves weakly capturing computations not of $\square^p_i$ functions, but of $\square^p_i[wit, O(\log n)]$ functions.

Because the oracle answers are witnessed, two computations may branch not only on a different oracle answer, but also by getting a different witness for a positive answer. So this time the correct computation is not unique, but that makes no difference in the proof other than the option of forgetting about the w's and using witnesses extracted from the computation instead.

The bound on the number of witnesses, namely $\big|\,|x|\,\big|$, allows to consider shorter tuples in place of ϵ, namely $\epsilon_1, \ldots, \epsilon_{k\cdot\big|\,|x|\,\big|}$ for some constant k with the same rôle in the proof.

The function to be minimized in $D(x)$, say $f(x,c)$ will be defined as $2^{k\cdot\big|\,|x|\,\big|} = 2^k \#|x|$ if $c \notin D_x$, and as 2^ϵ if $c \in D_x$. By 2^ϵ we mean taking ϵ as a $k \cdot \big|\,|x|\,\big|$-bit number and encoding it into unary; this is possible with a polynomial-time algorithm, because its input also includes the (unused) binary representation of x. ⊣

Lemma 6.5 *Let $i \geq 1$ and φ be a Σ^b_i-formula, $s(y)$ and $s'(x,y)$ terms. Then PV_1 together with the statements that $\#FM^{\Sigma^p_{i-1}}$ is well-defined proves, for some term t, the principle of sharply bounded collection:*

$$\forall x < |s|\ \exists z < s'\ \varphi(x,y,z) \iff \exists w < t(y)\ \forall x < |s|\ (w)_x < s' \wedge \varphi(x,y,(w)_x).$$

Proof: The proof is the same as in Lemma 5.9. A formal difference is that the function values (like $|s(y)| - i$) have to be replaced by their expansions into the unary. This is possible in polynomial time, because the resulting values are all bounded by $s(y)$. ⊣

Theorem 6.6 *Let $i \geq 2$. The search problems Σ^b_i-definable in S^{i-1}_2 are axiomatized by $\#FM^{\Sigma^p_{i-2}}$ search problems.*

Proof: The proof proceeds in analogy with the proof of Theorem 5.10. The desired fraction is axiomatized by Σ^b_{i-1}-length-minimization principles. Take any $\varphi(x) \in \Sigma^b_{i-1}$ and transform it to some $\exists y < T(x)\, \varphi_0(x,y)$ where $\varphi_0 \in \Pi^b_{i-2}$ as in Theorem 5.10, this time using Lemma 6.5.

Let the domain of the desired $\#FM^{\Sigma^p_{i-2}}$ problem be a set of pairs (x',y) where $x' \leq x$ and $y < T(x')$. Define

$$f_{\#FM}((x',y)) = \begin{cases} x' & \text{if } \varphi_0(x',y), \\ 2 \cdot T(x) & \text{otherwise.} \end{cases}$$

(The only difference from the proof of Theorem 5.10 is a "safety bit" in the unwanted case to avoid its length minimality in spite of non-minimality.) This time we get only one of the shortest (not necessarily smallest) minima, but this is enough to conclude the length minimization for φ. ⊣

Theorem 6.7 *The search problems Σ^b_1-definable in S^1_2 are captured by $\#FM$ search problems.*

Proof: The proof proceeds in full analogy with proof of Theorem 5.11, although again it is necessary to add "safety bits" in two places, to ensure that any shortest pair of some kind is guaranteed to be shortest in the "more important" coordinate, too. Technically this means multiplying by $2 \cdot t(x)$ instead of $t(x)$ and by the appropriate power of $2 \cdot M(u)$ instead of $M(u)$. ⊣

Chapter 7

$\Sigma_i^b(S_2^{i+1})$: The Subdiagonal

7.1 Polynomial Local Search

Definition 7.1 *(Johnson, Papadimitriou, Yannakakis [JPY88]) The* polynomial local search *problem (PLS) is defined by polynomial-time computable functions $N(x,y)$ and $v(x,y)$, for which*

$$v(x,N(x,y)) < v(x,y) \vee N(x,y) = y.$$

The search task is to find y such that $N(x,y) = y$.

Theorem 7.2 *(Buss, Krajíček [BK94]) Let $i \geq 1$. The class of $PLS^{\Sigma_{i-1}^p}$ problems weakly characterizes the Σ_i^b-consequences of S_2^{i+1}.*

The proof proceeds by a proof-theoretic analysis in S_2^{i+1} and can be found in [BK94]. A new proof of its base case $i = 1$ will be given in Section 8.2.

7.2 Generalized Local Search

Definition 7.3 *The* generalized local search *problem (GLS) is defined by a polynomial-time computable predicate $<(x,a,b,c)$ (conveniently always denoted as $a <_c b$) and a polynomial-time computable function $v(x,a)$ (conveniently denoted as $v(a)$) for which each $<_c$ is a linear order and*

$$(\forall d < D(x)\ \forall e < D(x)\ \neg d <_a b \wedge \neg e <_b c) \rightarrow v(c) \leq v(b).$$

The search task is to find a triple a, b and c such that b is $<_a$-minimal, c is $<_b$-minimal and $v(b) = v(c)$.

This definition is a simplified version of the one in [CK98]. The latter also uses an extra predicate $N(x, z, n)$, meaning "n is one of the neighbors of z", and each $<_z$ is required to be a linear order only over the set of neighbors of z — which is required to be nonempty for any z — instead of over the whole domain. Our definition is clearly a special case thereof, but it is actually many-one equivalent (and even under the *strong reduction* in the sense of [BCE+95]), as the order can be extended to non-neighbors by defining all non-neighbors to be greater than neighbors and by using an arbitrary linear order to compare non-neighbors among themselves.

Lemma 7.4 *Let η be a $PLS^{\Sigma_{i-1}^p}$ problem. There is a $GLS^{\Sigma_{i-2}^p}$ problem η' such that*

$$S_2^1 \vdash \forall x\, \forall s\, (\eta'(d'(x), s) \rightarrow \eta(x, g'(x, s)))$$

where d' and g' are polynomial-time computable functions.

Proof: Suppose η consists of $N_\eta(x, y)$ and $v_\eta(x, y)$ as in the definition of PLS.

Take M to be a Turing machine with an oracle from Σ_{i-1}^p which computes simultaneously $N_\eta(x, y)$ and $v_\eta(x, y)$ in time $(|x| + |y|)^m$. Fix x and define D as the set of sextuples of the form:

$$(N, v, p, w, \epsilon, y)$$

where

1. p is a computation of machine M with input (x, y) and output (N, v) where $y, N < D(x)$
2. ϵ is a tuple $\epsilon_1, \ldots, \epsilon_{|x|^m}$ of oracle answers received by the computation p (in order; padded with zeroes at the end)
3. w is a tuple $w_1, \ldots, w_{|x|^m}$ of witnesses to positive oracle answers received by the computation p (so that w_n corresponds to $\epsilon_n = 1$; padded with zeroes elsewhere)

We say that $d \in D$ is a *sequel* to e if $(e)_1 = (d)_6$, i.e., the PLS-style neighbor suggested in e is used as the input in d.

The $GLS^{\Sigma_{i-2}^p}$ relation $d <_z e$ is defined as always true if d is a sequel to z, while e is not. It is defined antilexicographically by the respective values

of ϵ if both are a sequel to z and it is defined lexicographically by the whole sextuples in the less important cases — when none is a sequel to z, or when both are a sequel to z and $(d)_5 = (e)_5$.

The $GLS^{\Sigma_{i-2}^p}$ value of any $(N, v, p, w, \epsilon, y)$ is the v component. More formally, $v_{\eta'}(d) = (d)_2$.

Argue in S_2^1: The set D is nonempty: take any y, assume all oracle answers to be negative, all witnesses w to be zero and apply Fact 2.15 to construct p, N and v from y and $\epsilon = (0, \ldots, 0)$ as defined. By the same argument, for any z, there is a d which is a sequel to z: just start with $y = (z)_1$. Therefore, every $<_z$-minimum is a sequel to z and is determined by the ϵ-maximality. By the same argument as in Theorem 5.8 we get the correctness of p with respect to the oracle and so the promise of the underlying $PLS^{\Sigma_{i-1}^p}$ problem η translates to the characteristic promise of the $GLS^{\Sigma_{i-2}^p}$ problem.

Any solution

$$(N_a, v_a, p_a, w_a, \epsilon_a, y_a),\ (N_b, v_b, p_b, w_b, \epsilon_b, y_b),\ (N_c, v_c, p_c, w_c, \epsilon_c, y_c)$$

to the $GLS^{\Sigma_{i-2}^p}$ problem satisfies $N_b = N_\eta(x, y_a)$, $v_b = v_\eta(x, N_b)$ and $N_c = N_b$. Hence N_c is a solution to η. ⊣

Theorem 7.5 *Let $i \geq 2$. The class of $GLS^{\Sigma_{i-2}^p}$ problems weakly characterizes the Σ_i^b-consequences of S_2^{i+1}.*

Proof: The definition of $GLS^{\Sigma_{i-2}^p}$ is slightly complicated, but rewritten into a formula $\varphi(x, y)$, where $y = (a, b, c)$, it turns out to be a Σ_i^b-formula[1].

It is definable in $T_2^i \subseteq S_2^{i+1}$: take x as a constant and consider the set

$$\left\{v_b : \exists a < D(x)\ \exists b < D(x)\ \forall d < D(x)\ \neg d <_a b \wedge v(b) = v_b\right\}$$

which is thus defined by a Σ_i^b-formula with the only free variable being v_b. It is also nonempty, because T_2^i proves each $<_a$ has a minimum. By the Σ_i^b-minimization principle available in T_2^i, this set has a minimum, yielding a and b. Take c to be the $<_b$-minimal element. The triple (a, b, c) is a solution to the $GLS^{\Sigma_{i-2}^p}$ problem.

[1]To weakly characterize the class of $GLS^{\Sigma_{i-2}^p}$ problems by a Π_{i-1}^b-formula, take the "overdone no-promise" version of GLS instead.

Take any Σ_i^b-formula $\theta(x,y)$ and the corresponding $PLS^{\Sigma_{i-1}^p}$ problem η to which it is reducible by Theorem 7.2:

$$\forall x\, \forall s\, (\eta(d(x),s) \to \theta(x,g(x,s))).$$

Lemma 7.4 provides a $GLS^{\Sigma_{i-2}^p}$ problem η' such that

$$S_2^1 \vdash \forall x\, \forall s\, (\eta'(d'(x),s) \to \eta(x,g'(x,s)))$$

and the weak characterization follows by the transitivity of reductions. $\dashv$

It would be a convenient strengthening if Theorem 7.2 could provide an η where

$$PV_1 \vdash \forall x\, \forall s\, (\eta(d(x),s) \to \theta(x,g(x,s)))$$

but such result is not available and known methods prove the reduction, even for the base case, in S_2^2, or at most in T_2^1. We shall however return to what can be said about GLS and $\Sigma_2^b(S_2^3)$ after additional methods are developed, in Corollary 9.4 and Theorem 9.6.

7.3 Minimization

Definition 7.6 *The search problem $GLS^=$ is a special case of GLS with the cost being an arbitrary constant.*

To satisfy the reader who is accustomed to the old style definition of GLS found in [CK98], we should also briefly investigate the old style definition of $GLS^=$, which is temporarily reintroduced as $GLS_N^=$. The expected confirmation of the many-one equivalence is reached again, though slightly less directly this time.

Definition 7.7 *The search problem $GLS_N^=$ is a special case of the old style GLS, with the cost being constant over the whole neighborhood of each z. More formally: the problem is defined by predicates $a <_c b$ and $N(c,a)$ (meaning "a is a member of the neighborhood of c"), and a function $v(a)$; all three depend also on x. The promised transitivity, antisymmetry and linearity of $<_c$ is restricted (weakened) to the neighborhood of c. The GLS-specific*

promise is also weakened by additional premises $N(a,b)$ and $N(b,c)$. Further two promise conjuncts are given:

$$\forall a < D(x)\ \exists b < D(x)\ N(a,b)$$

$$\forall c,a,b < D(x)\ \big((N(c,a) \wedge N(c,b)) \rightarrow v(b) = v(c)\big)$$

The search task is as in GLS, but with the additional requirements that $N(a,b)$ and $N(b,c)$.

We are being honest here; if we required the cost being constant not only over every neighborhood, but over all of the domain[2], we would again easily observe many-one equivalence with $GLS^{=}$. That would only strengthen our claim that the neighborhood predicate in the old style definition never makes any difference.

Note that MIN as defined in the Section 3.3 is actually a special case of $GLS^{=}$ with $<_z$ being independent of z. But even more:

Lemma 7.8 $MIN \approx_m GLS_N^{=} \approx_m GLS^{=}$

Proof: Show the reducibility of $GLS_N^{=}$ to MIN by constructing a linear order over triples of elements of any instance of $GLS_N^{=}$.

Define $(a,b,c) <^L (a',b',c')$ to be true, if one of the following holds:

$$\begin{aligned}
&N(a,b) \wedge \neg N(a',b')\\
&N(a,b) \wedge N(a',b') \wedge N(b,c) \wedge \neg N(b',c')\\
\neg&N(a,b) \wedge \neg N(a',b') \wedge (a,b,c) < (a',b',c')\\
\neg&N(b,c) \wedge \neg N(b',c') \wedge (a,b,c) < (a',b',c')
\end{aligned}$$

or if all $N(a,b)$, $N(b,c)$, $N(a',b')$ and $N(b',c')$ and one of the following hold:

$$\begin{aligned}
&v(b) < v(b')\\
&v(b) = v(b') \wedge a < a'\\
&v(b) = v(b') \wedge a = a' \wedge b <_a b'\\
&v(b) = v(b') \wedge a = a' \wedge b = b' \wedge c <_b c'.
\end{aligned}$$

This is a linear order and its unique minimum is some (a,b,c) which satisfies the search task of $GLS_N^{=}$. In particular, the value $v(b)$ is globally

[2]Remember that $GLS^{=}$ did not receive attention before, and so we have to resort to some kind of analogy to identify the old style definition.

minimal among the values which can occur in a neighborhood of any element, and so $v(b) \leq v(c)$, which, by the promise, means $v(b) = v(c)$.

The rest of the lemma is trivial as MIN is a special case of $GLS^{=}$, which is a special case of $GLS^{=}_{N}$. ⊣

It is open whether MIN is many-one equivalent (or at least Turing equivalent) with a more general search problem MIN^P of finding a minimum of a partial order, i.e., MIN with the linearity component of the promise dropped.

It is also open whether MIN is many-one equivalent with $\underset{NOPROMISE}{MIN}$.

As all these search problems are in $\Sigma_2^b(S_2^3)$, any negative result concerning these main variations of the MIN problem would also give a negative answer to a question asked in [CK98]: does MIN (weakly) characterize $\Sigma_2^b(S_2^3)$?

Let us conclude this section by three partial results helping to understand the relation between MIN and $\underset{NOPROMISE}{MIN}$.

Lemma 7.9 *There is a Turing reduction of* $\underset{NOPROMISE}{MIN}$ *to* MIN *with only two queries to the* MIN *oracle, both coming from a set of queries constructed before any oracle answers are received.*

Proof: It is possible to resolve any Σ_1^b-question, i.e., an arbitrary NP question, using MIN: simply define witnesses to be smaller than non-witnesses and use some arbitrary linear order to compare witnesses among themselves and non-witnesses among themselves.

The first query of the Turing reduction asks the Σ_1^b-question whether there is a triple of elements of the domain of the $\underset{NOPROMISE}{MIN}$ problem being reduced, which breaks at least one of the transitivity, antisymmetry and linearity. If so, the reduction can halt with a correct output. Otherwise a second query is asked which is identical to the to the problem instance being reduced, because it is now certain that it happens to be a linear order. ⊣

This result is complemented by the following one (for $R = \underset{NOPROMISE}{MIN}$).

Lemma 7.10 *Let* R *be a search problem. Suppose there is a Turing reduction of* R *to* MIN *such that the reduction algorithm computes some* n*, builds an algorithm* $C(y_1, \ldots, y_n)$ *and* n *queries to the oracle (*each satisfying the promise of MIN*) without actually asking any, then asks all the queries, evaluates* C *on the oracle answers and halts with the output of* C*. Then* $R \preceq_m MIN$.

Proof: Call the reduction as described above *conjunctive truth table reduction.* Many-one reducibility is a special case of conjunctive truth table reducibility with $n = 1$.

Assume there is a conjunctive truth table reduction of R to MIN. Then n used in the reduction is a polynomial-time computable function of the input and the length of any oracle query is bounded by a polynomial t_q. The i-th oracle query consists of a linear order $<_i$ and an instance d_i, $1 \leq i \leq n$, and the oracle answer will eventually be some y_i.

Define a new $<^L$ in the domain $d = t_q \# n$, on members of this form:

$$x = (x_1, \ldots, x_n) \qquad x_i < d_i,$$

lexicographically:

$$x <^L x' \iff \exists i < |d|\ x_1 = x'_1 \wedge \cdots \wedge x_{i-1} = x'_{i-1} \ \wedge\ x_i <_i x'_i.$$

This is a linear order; its unique minimum is some $(m_1, \ldots, m_n)$. Finish the many-one reduction of R to this instance of MIN by halting with output $C(m_1, \ldots, m_n)$. $\dashv$

The two lemmata just given hold analogously for search problems related to partial orders MIN^P and $\underset{NOPROMISE}{MIN^P}$, which we are now going to define.

Definition 7.11 *The search problem MIN^P is defined by a ternary predicate $<^P(a, x, y)$ conveniently denoted as $x <^P y$, such that for any a, $<^P$ is a partial order and the search task is to find a minimum element of $<^P$.*

Definition 7.12 *The search problem $\underset{NOPROMISE}{MIN^P}$ is again defined by a ternary predicate $x <^P y$. There is no promise and the search task is to find a triple which breaks the transitivity of $<^P$, a pair which breaks the antisymmetry of $<^P$, or an element which is a minimum element of $<^P$.*

Definition 7.13 *The search problem $\#MIN^P$ is a search problem defined like MIN, but instead of the linearity assume a polylogarithmic bound on the depth of the order in the promise; the polylogarithmic bound is supplied through a function symbol. More formally, the extraarithmetical symbols are $<^P$ (ternary) and b (unary); the promise is the transitivity of $<^P$, the antisymmetry of $<^P$ and the nonexistence of a set of size $|b|$ (the* bound*) which is linearly ordered by $<^P$; the search task is to find any minimum element.*

Definition 7.14 *The search problem* $\#MIN^P_{\text{NOPROMISE}}$ *has no promise and its search task is to find a counterexample to transitivity or antisymmetry, a linearly ordered set of size* $|b|$*, or a minimum.*

Lemma 7.15 *The* $\#MIN^P_{\text{NOPROMISE}}$ *and* $\#MIN^P$ *search problems are many-one equivalent.*

Proof: The proof of $\#MIN^P_{\text{NOPROMISE}} \preceq_m \#MIN^P$ is based on the observation that each linearly ordered subset of a partially ordered set either includes a global minimum, or can be extended to a larger one.

The reduction takes an input with domain d and bound $|b|$ (for simplicity, assume $|b| \geq 3$), and asks a query with bound $|b'| = |b|+4$ (i.e., $b' = 16 \cdot b$) and the order to be minimized is the opposite of the cardinality-based order of linearly ordered sets of at most b elements. More formally: the domain is an upper limit on the codes depending on the chosen coding of sets, for example $d\#d$. The members of the domain are classified into four groups according to the size and the behavior of the input "order" on the set encoded by the member:

1. codes of non-transitive or non-asymmetric sets
2. codes of transitive, asymmetric and linear sets of size $|b|+1$ or more
3. codes of transitive, asymmetric and linear sets of size at most $|b|$
4. codes of transitive and asymmetric, but non-linear sets, and non-codes

The queried order $u <_q v$ is true either if the group of u is lower numbered than the group of v, or if both u and v are of group 3 and v has fewer elements than u. This indeed defines an order of depth at most $|b|+4$ no matter what the input relation is.

If transitivity or antisymmetry has been violated in the input, the minimum for the query is of group 1 and any such minimum provides enough information to compute the output of the reduction. If this has not been the case, but the input has breached its bound $|b|$, the minimum is a set of group 2 and it serves as the output of the reduction. If the input has satisfied all of the promise, then the minimum is of group 3, and its minimum element serves as the output of the reduction. $\dashv$

7.4 Generalized Iteration

Definition 7.16 *The search problem* $\underset{NOPROMISE}{GI}$ *is defined by a ternary relation* $<_{GI}(a,x,y)$ *(conveniently denoted as* $x <_{GI} y$*; possibly a linear order in the indicated variables) and a binary function* $g(a,x)$ *(conveniently denoted as* $g(x)$*). The problem has no promise and the search task is to identify a member which breaks one the following restrictions on* g*:*

$$\begin{aligned} &g(0) <_{GI} 0, \\ &g(x) <_{GI} x \rightarrow g(g(x)) <_{GI} g(x), \\ &g(x) < D(x), \end{aligned}$$

or a pair or a triple of elements which breaks the linearity, transitivity or antisymmetry of $<_{GI}$*.*

Definition 7.17 *The search problem* $\underset{PROMISE}{GI}$ *is defined in the spirit of* $\underset{NOPROMISE}{GI}$*, while its promise is that* $<_{GI}$ *is a linear order and* $g(0) <_{GI} 0$*. The search task is to find* x *such that* $g(x) <_{GI} x$*, but not* $g(g(x)) <_{GI} g(x)$*.*

Definition 7.18 *The search problem* MIN^* *is defined by a ternary predicate* $<^L (a,x,y)$ *conveniently denoted as* $x <^L y$ *and a binary function* $h(a,x)$ *conveniently denoted as* $h(x)$*. The promise is that* $<^L$ *is a linear order.*

The search task is to find y *such that* $\neg h(y) <^L y$*.*

The notation requires some explanation. The problems GI and MIN were defined in [CK98] as candidates for the weak characterization of Σ_1^b- and Σ_2^b-consequences, respectively, of T_2^2. (Those questions are still open.) MIN^* is defined from MIN by a specific type of Herbrandization: in addition to the linear order a device to derive an example of a smaller element for any element is available. This Herbrandization will be explained in a more general setting in Chapter 8.

An attentive reader will also note a difference between our definition of $\underset{PROMISE}{GI}$ and the definition of GI introduced in [CK98], in the choice of the search task. Whereas the original GI includes

$$g(x) <_{GI} x \rightarrow g(g(x)) <_{GI} g(x)$$

as a part of the promise and falsifies $g(x) < D(x)$ in the search task, we have chosen the opposite for $\underset{PROMISE}{GI}$. The reason is a clearer affinity to the other well-known problems we study, and a more transparent road to Corollary 7.25. Of course, the original GI and our $\underset{PROMISE}{GI}$ are actually many-one equivalent, too. Both directions of the proof are similar to each other and easy, as the affected conditions can be locally verified in polynomial time: after the necessary promise of the problem reduced to is satisfied by local changes where necessary, only the $g(0) <_{GI} 0$ part of the promise may become breached. It is then regained by the technique of Lemma 7.23 below. The same technique also addresses another non-essential variation: the promise may or may not include a statement that 0 is the maximum element of $<_{GI}$.

Definition 7.19 *The search problem GLS' is defined in the same way as the GLS search problem, but an additional function $f(x, z, y)$ (conveniently denoted as $f(z, y)$) is available with an additional promise:*

$$f(z, y) <_z y \vee \big(f(z, y) = y \wedge \forall u < D(x)\ (y = u \vee y <_z u)\big)$$

(i.e., identity on minima, elsewhere a witness to the non-minimality).

We shall write $\min_z y$ to abbreviate $f(x, z, y) = y$ (which is equivalent to the fact that the minimum of $<_z$ is y).

At the first sight the relation of MIN^* to MIN on one hand and of GLS' to GLS on the other hand looks rather similar — and rightfully so, as the remaining difference of putting the "Herbrandizing" behavior of the additional function either to the promise or to the search task does not affect the many-one equivalence about to be reached in Corollary 7.25, either. The notation does not reflect this apparent parallel; the star notation has to stay reserved for a much more general construction developed in Chapter 8. Taking into account Lemma 7.8 and Corollary 7.25, GLS' would be a poor counterpart for MIN^* in direction of GLS.

Definition 7.20 *The search problem $GLS'^{=}$ is the special case of GLS', where the cost is an arbitrary constant.*

Start up with a trivial observation even though it will be immediately replaced by a couple of slightly stronger results with slightly more complicated proofs.

Lemma 7.21 $MIN^* \preceq_m GLS'^{=}$.

Proof: Define the $x <_z y$ as $x <_{MIN^*} y$ (independently of z, but still dependent on a as always), $f(z,x)$ as $h(x)$, and let the cost be an arbitrary constant. $\dashv$

Lemma 7.22 $\underset{PROMISE}{GI} \preceq_m GLS'^{=}$.

Proof:

If $g(x) < x \wedge g(y) < y$, define $x <_z y$ as $x <_{GI} y$ as above

If $g(x) \geq x \wedge g(y) \geq y$, define $x <_z y$ as $x < y$

If $g(x) \geq x \wedge g(y) < y$, let $x <_z y$ be false

If $g(x) < x \wedge g(y) \geq y$, let $x <_z y$ be true

It is easy to see that this defines a linear order, as it is a "concatenation" of two linear orders, namely of $<_{GI}$ and of the lexicographic order $<$. (Instead of the lexicographic order, any linear order would do, too). It remains to define f so as to be decreasing in this order, except for the $<_z$-minimal element:

If $g(g(x)) < g(x) < x$, define $f(z,x)$ as $g(x)$

If $g(x) \geq x$, define $f(z,x)$ as 0

If $g(g(x)) \geq g(x) \wedge g(x) < x$, define $f(z,x)$ as x.

In the middle item the promise that $g(0) <_{GI} 0$ is used.

By the same promise, any minimum x of any $<_z$ found by the query to the $GLS'^{=}$ oracle has the property that $g(x) < x$ holds. By its $<_z$-minimality, $g(x)$ lacks this property and so x is a suitable output of the reduction. $\dashv$

Lemma 7.23 $MIN^* \preceq_m \underset{PROMISE}{GI}$.

Proof: The domain of $\underset{PROMISE}{GI}$ is the domain of MIN^* increased by one.

Define $0 <_{GI} x$ to be always false and $x+1 <_{GI} 0$ always true; define $x+1 <_{GI} y+1$ as $x <^L y$. Define $g(x+1)$ as $h(x)$ and $g(0) = 1$. This satisfies the promise of $\underset{PROMISE}{GI}$ and the search task of $\underset{PROMISE}{GI}$ is actually stronger than that of MIN^*. $\dashv$

Lemma 7.24 $GLS' \preceq_m MIN^*$

Proof: The domain of the MIN^* problem will consist of triples of the GLS' problem elements.

The ordering predicate $(x,y,z) <_{MIN^*} (x',y',z')$ will be true in any of the following eight cases:

1 $\min_x y \wedge \neg \min_{x'} y'$

2 $\min_x y \wedge \min_{x'} y' \wedge v(y) < v(y')$

3 $\min_x y \wedge \min_{x'} y' \wedge v(y) = v(y') \wedge y < y'$

4 $\min_x y \wedge \min_{x'} y' \wedge y = y' \wedge z <_y z'$

5 $\min_x y \wedge \min_{x'} y' \wedge y = y' \wedge z = z' \wedge x < x'$

6 $\neg \min_x y \wedge \neg \min_{x'} y' \wedge x < x'$

7 $\neg \min_x y \wedge \neg \min_{x'} y' \wedge x = x' \wedge y <_x y'$

8 $\neg \min_x y \wedge \neg \min_{x'} y' \wedge x = x' \wedge y = y' \wedge z <_y z'$

(The reader is reminded that the $<$ predicate without any further qualification is a standard arithmetical predicate, here used both for numeric comparison of values and for the "lexicographic" comparison of other objects. Any other polynomial-time order could perform its job of choosing a unique value-minimal object in the latter context as well.)

To see that the just defined predicate is indeed a linear order, note that it is actually a hierarchy of value-based partial orders, hence antisymmetry and transitivity. Linearity is obvious.

The iterator function $g((x,y,z))$ is accordingly defined by cases.

If $\neg \min_x y$, then $g((x,y,z)) = (x, f(x,y), 0)$

If $\min_x y \wedge \neg \min_y z$, then $g((x,y,z)) = (x, y, f(y,z))$

If $\min_x y \wedge \min_y z$, then $g((x,y,z)) = (y, z, 0)$

Check that for every x, y, and z, $g((x,y,z)) <_{MIN^*} (x,y,z)$.

If $\neg \min_x y$, either case 1 or 7 occurs.

If $\min_x y \wedge \neg \min_y z$, case 4 occurs.

If $\min_x y \wedge \min_y z$, case 2 occurs.

⊣

Corollary 7.25 $GLS'^{=} \approx_m GLS' \approx_m \underset{PROMISE}{GI} \approx_m MIN^*$

Analogous definitions of the "overdone no-promise versions" together with analogous reasoning give an analogous result.

Corollary 7.26 $\underset{NOPROMISE}{GLS'^{=}} \approx_m \underset{NOPROMISE}{GLS'} \approx_m \underset{NOPROMISE}{GI} \approx_m \underset{NOPROMISE}{MIN^*}$

It is natural to ask whether the last two corollaries can be combined by a many-one equivalence, too. We do not know. This is related to the $\underset{NOPROMISE}{MIN} \approx_m MIN$ question and to Lemma 7.10. The same picture also appears if we simply omit the linearity promise from all the search problems linked by the results of this section.

Chapter 8

Star Herbrandization

We will now present a general method of transferring known characterizations downwards to fractions. The idea is simple: let us start from a Σ^b_2-definable search problem X which characterizes the Σ^b_2-consequences of some theory. We want to derive, say, a characterization of the Σ^b_1-consequences of the same theory. The method is to weaken X by adding an extra function symbol h to it, which provides witnesses of a solution candidate not being a solution. This function symbol serves to remove one level of quantification when the search problem is formulated. It turns out that the strength of X with regard to Σ^b_1-formulae is preserved, if the system of all such weakenings is taken.

The name "star Herbrandization" was chosen to distinguish this technique from the "dagger Herbrandization" discussed in Chapter 9.

8.1 General Construction

Definition 8.1 *Let X be a class of search problems uniformly defined by a strictΣ^b_2-formula*

$$\exists y' < D(x)\, \forall z < D(x)\ \ \varphi(x, y, y', z)$$

in language $L_2(\alpha, \dots)$ for φ sharply bounded; i.e., the search task is to find y such that $\varphi(x, y, y', z)$ holds for a suitable y' and all relevant z. Its star Herbrandization *X^* is the Σ^b_1-definable search problem in language $L_2(\alpha, \dots, h)$ with the search task to find y such that*

$$\exists y'\, h(x, y, y') < D(x) \to \varphi(x, y, y', h(x, y, y')).$$

The limitation to *strict* Σ^b_2-formulae, as opposed to Σ^b_2-formulae, is not a serious one. All Σ^b_2-definable search problems investigated until now can be easily expressed in this form. Also, the following arguments can be extended a little bit (using the result of [Res86] accessible as [Kra95, Lemma 5.2.13], and straightforward subformula manipulation) to all Σ^b_2-formulae in which no sharply bounded universal quantifiers appear before or inside the initial block of bounded existential ones.

Theorem 8.2 *Let $X \in strict\Sigma^b_2$ and $Y \in strict\Sigma^b_1$ be formulae uniformly defining their respective classes of search problems. If X captures Y, then X^* captures Y, too.*

(The converse implication is obvious.)
Proof: Suppose X has the form as in Definition 8.1. By the capture,

$$PV_1(\alpha) \vdash \forall x\, \forall v\, (\exists y < D(x)\, \forall z < D(x)\, \varphi(f(x), v, y, z, \beta) \rightarrow Y(x, g(x, v), \alpha)),$$

or equivalently

$$PV_1(\alpha) \vdash \forall x\, \forall v\, \forall y < D(x)\, \exists z < D(x)\, (\varphi(f(x), v, y, z, \beta) \rightarrow Y(x, g(x, v), \alpha))$$

with φ sharply bounded.

Apply the Herbrand's theorem to obtain a finite number of terms — polynomial-time functions, which provide alternatives for witnessing both the existence of such z and the initial existential quantifier of Y. As the rest of the parenthesized subformula is sharply bounded, these terms can be combined into a single polynomial-time function $h(x, v, y)$, which witnesses the existence of a z as stated. Formally:

$$PV_1(\alpha) \vdash \forall x\, \forall v\, \forall y < D(x)\, (X(f(x), v, y, h(x, v, y), \beta) \rightarrow Y(x, g(x, v), \alpha)),$$

which is an instance, for this particular h, of

$$PV_1(\alpha) \vdash \forall x\, \forall v\, \forall y < D(x)\, (X^*(f(x), v, y, \beta) \rightarrow Y(x, g(x, v), \alpha)).$$

by the definition of X^*. ⊣

Theorem 8.2 can be replaced by the following more general theorem that does not require Y to be $strict\Sigma^b_1$. This is paid by an indirect dependence on Buss' theorem — which renders it unusable for Corollary 8.6.

Theorem 8.3 *Let $X \in strict\Sigma^b_2$ and $Y \in \Sigma^b_1$ be formulae uniformly defining their respective classes of search problems. If X captures Y, then X^* captures Y, too.*

Proof: As the proof of 8.2. The problematic step is the combination of alternative witnessing terms; the construction of the proof covers initial bounded existential quantifiers and the sharply quantified kernel, but not additional bounded existential quantifiers interspersed with sharply bounded ones. It is thus necessary to weaken the statement of the capture to $S^1_2(\alpha)$ and to use the sharply bounded collection for Σ^b_1-formulae, which is provable in $S^1_2(\alpha)$, to replace a block of bounded existential quantification by a single quantifier. Having obtained

$$S^1_2(\alpha) \vdash \forall x\, \forall v\, \forall y < D(x)\, \exists z < D(x)\, (\varphi(f(x), v, y, z, \beta) \rightarrow Y(x, g(x, v), \alpha)),$$

apply the Buss' theorem and combine the terms. To return back to the weaker PV_1, the Σ^b_1-conservativity of $S^1_2(\alpha)$ over $PV_1(\alpha)$, which is a direct consequence of the relativized Buss' theorem, is used. $\dashv$

Corollary 8.4 *Suppose X is a $strict\Sigma^b_2$-definable class of search problems which characterizes the Σ^b_2-consequences of a theory T, $PV_1 \subseteq T$. Then X^* characterizes the Σ^b_1-consequences of T.*

Proof: The well-definedness of X^* is a direct consequence of the well-definedness of X. Its bounded quantifier complexity is Σ^b_1. X^* captures all Σ^b_1-consequences of T by the theorem. $\dashv$

8.2 Applications

One application of the results of this chapter was already presented separately: the connection between $\underset{PROMISE}{GI}$ and MIN^* as stated and proved in Corollary 7.25.

A simple new proof of the basic case of Theorem 7.2 and even of Buss' theorem (Theorem 5.2) can also be obtained as a corollary of Theorem 8.2. These proofs are not different in the sense that they would not ultimately

rely on proof-theoretic analysis. They do, but the results depended upon are simpler than the quantifier elimination needed for the original proof of Buss. They rely on Theorem 6.1 and also on the witnessing theorem from [KPT91]. On the other hand, the proofs therefore show a new connection linking together Theorem 7.2, Theorem 5.2 and Theorem 6.1.

Corollary 8.5 *The PLS problems characterize the strictΣ^b_1-consequences of S^2_2.*

Proof: Assume $S^2_2 \vdash \forall u\, \exists v\, \theta(u,v)$, where θ is a *strict*Σ^b_1-formula. By Theorem 5.11, FM captures the class of search problems uniformly defined by θ. By Theorem 8.2, FM^* captures the same class, too. To conclude the proof, it is sufficient to observe $FM^* \preceq_m PLS$ within PV_1; namely, the function to be minimized f_{FM} can be identified with the cost v_{PLS} of a PLS problem, and the function h introduced by the star Herbrandization can be trivially adapted to a neighborhood function N_{PLS}, which satisfies the promise of the PLS problem, and whose fixed points are exactly the solutions to the underlying FM^* search problem. ⊣

(Use Theorem 8.3, which does use the Buss' theorem, in place of Theorem 8.2 which does not, to extend the corollary to all Σ^b_1-consequences of S^2_2.)

Corollary 8.6 *The class of $\Box^p_1$ functions taken as search problems characterizes the strictΣ^b_1-consequences of S^1_2.*

Proof: Assume $S^1_2 \vdash \forall u\, \exists v\ \ \theta(u,v)$, where θ is a *strict*Σ^b_1-formula. By Theorem 6.7, $\#FM$ captures the class of search problems uniformly defined by θ. By Theorem 8.2, $\#FM^*$ captures the same class, too. To conclude the proof, it is sufficient to present a reduction of $\#FM^*$ to a $\Box^p_1$ function taken as a search problem, which is provable in PV_1. We can even present a reduction to the identity function using the following polynomial-time function:

Given x, compute the known upper bound on maximum running time t of $f_{\#FM}$ for inputs up to x. This is also an upper bound on output sizes of $f_{\#FM}$. Build a value m such that $|m| > t$ using Fact 2.15. Iterate the function h (starting from, e.g., zero) introduced by the star Herbrandization t times and output the result. Use the PV_1-specific variant of polynomial induction for the formula expressing that the i-th iteration provides a value shortened by at least i bits to prove the reduction in PV_1. ⊣

It is a natural question whether this corollary can be extended to all Σ^b_1-consequences of S^1_2 without getting into a vicious circle by using the Buss' theorem indirectly. A seductive device is Lemma 6.5 and it is almost enabled by the concrete X for which Theorem 8.3 is reproved without the excursion to $S^1_2(\alpha)$. The difficulty of its application is that a particular $f(x)$ and a particular β is available, whereas the sharply bounded collection instance needed has x in place of $f(x)$ and a different oracle for f_{FM} than β. The different oracle can be made available at some expense of clarity. Just add the required instance of the FM search problem to the finite set considered for a specific Σ^b_1-formula φ at the beginning of the proof of Theorem 5.11. The construction by which the instances of FM are combined is reversible and we can recover this additional instance in the proof of Theorem 8.3. Unfortunately we have found no similar argument for the instance x and so this particular question remains unresolved. Fortunately, all Σ^b_1-definable search problems studied to our knowledge anywhere, are also $strict\Sigma^b_1$-definable.

Chapter 9

Dagger Herbrandization

Now we introduce a more general, but also a less straightforward method of obtaining new axiomatizations from the existing ones.

9.1 General Construction

A notational convention: by a Σ^b_2*-formula with no sharply bounded quantification* we now understand a Σ^b_2-formula with no sharply bounded quantifier such that there is other quantification than sharply bounded within its scope. We do this to be able to write a single quantifier instead of a vector of quantifiers of the same type. In this way we even combine the existential quantifier which asserts the existence of solutions with the initial existential quantifiers to keep the presentation as readable as possible.

Definition 9.1 *Let X be a search problem definable by a Σ^b_2-formula with no sharply bounded quantification, whose well-definedness is expressed (see Lemma 3.6) as*

$$\forall x\, \exists y < D(x)\, \forall z < D(x)\ \ \varphi(x, y, z),$$

where φ is sharply bounded; i.e., the search task is to find y such that $\varphi(x, y, z)$ holds for all relevant z, and both y and z encode as many variables of the indicated quantification type as necessary.

Its dagger Herbrandization $X^\dagger$ *is a class of search problems with the search task to find $w = (y_1, \ldots, y_k)$ with the property*

$$\varphi(t_1(x), y_1, s_1) \wedge \varphi(t_2(x, y_1), y_2, s_2) \wedge \cdots \wedge \varphi(t_k(x, y_1, \ldots, y_{k-1}), y_k, s_k),$$

where k is an arbitrary number, each s_i and t_i is an arbitrary polynomial-time function, and where every s_i may depend on x and all y_j, whereas t_i only on the variables shown.

An inconspicuous property of this construction is that the resulting $X^\dagger$ never has any promise. This is not surprising. A reasonable way to identify a possible promise of a search problem $\varphi(x, y)$ is that it is a disjunct which does not depend on y. But as all the disjuncts usually depend on x, and we have replaced x with functions which *do* depend on y, and also have covered the original formula by a conjunction, there is no opportunity for having a promise anymore. It follows that, e.g., $MIN^\dagger$ and $\underset{NOPROMISE}{MIN}^\dagger$ are the same class[1] of search problems, no matter whether MIN and $\underset{NOPROMISE}{MIN}$ are many-one equivalent or not.

Note that $X^\dagger$ is a class of definable search problems, but in contrast with all preceding constructions, it does not appear to be uniformly definable, even if X is uniformly definable; in the following theorem, Σ^b_1-consequences with longer proofs may need an instance of $X^\dagger$ with a higher k.

Theorem 9.2 *Let X be a union of classes of search problems definable by Σ^b_2-formulae with no sharply bounded quantification. The classes of strictΣ^b_1-consequences of X and of $X^\dagger$ over PV_1 are the same.*

Proof: Understand X as a set of formulae specifying the well-definedness of all search problems in X. Take any $\exists v' < D(u)\ \psi(u, v, v')$ which is a consequence of X, and thus also of some finite $X' \subseteq X$. Make sure that variables found in different members of X' are distinct. Join the formulae so as to receive a single Σ^b_2-formula, and, after straightforward finite sequence encoding, even a formula (whose well-definedness still implies all members of X' over PV_1) of the form

$$\forall z < D'(x)\ \varphi(x, y, z),$$

and observe

$$PV_1 \vdash \forall x\ \exists y\ \forall z{<}D'(x)\ \varphi(x, y, z) \rightarrow \forall u\ \exists v\ \exists v'{<}D(u)\ \psi(u, v, v').$$

[1]Which is, by the way, a candidate for the axiomatization of $\Sigma^b_1(S^3_2)$ at least as good as MIN is for $\Sigma^b_2(S^3_2)$, by Theorem 9.2.

From now on, treat u as a constant and apply Lemma 3.6 to extend the domain of X to $D''(x)$. That allows

$$PV_1 + \forall v\, \forall v'{<}D(u)\, \neg\psi(u,v,v') \vdash \exists x\, \forall y < D''(x)\, \exists z < D''(x)\, \neg\varphi(x,y,z).$$

Observe that the indicated theory in language $L_{PV} \cup \{u\}$ is a universal one. That allows to apply the Herbrand's theorem from [KPT91]. This yields k, l, and terms — polynomial-time functions t_i and s_i^j such that:

$$\begin{aligned} PV_1 + \forall v\, \forall v'{<}D(u)\, \neg\psi(u,v,v') \vdash\ & \forall y_1, \ldots, y_k < D''(x) \\ & \neg\varphi(t_1(u), y_1, s_1^1) \vee \ \ldots \ \vee \neg\varphi(t_1(u), y_1, s_1^l) \vee \\ & \neg\varphi(t_2(u, y_1), y_2, s_2^1) \vee \ \ldots \ \vee \neg\varphi(t_2(u, y_1), y_2, s_2^l) \vee \\ & \qquad\qquad \vdots \\ & \neg\varphi(t_k(u, y_1, \ldots, y_{k-1}), y_k, s_k^1) \vee \ \ldots \ \vee \neg\varphi(t_k(u, y_1, \ldots, y_{k-1}), y_k, s_k^l). \end{aligned}$$

(While the t_i's depend only on the variables shown, any s_i^j may depend on all u, $y_1, \ldots, y_k$.)

Now for each s_i^1 to s_i^l a single polynomial-time computable function s_i can be defined by cases in order to replace every line of the disjunction above by

$$\neg\psi(t_i(u, y_1, \ldots, y_{i-1}), y_i, s_i).$$

(Each of these s_i^j's is computable in polynomial time, and polynomial time is also sufficient for determining whether the sharply bounded formula ψ holds for given polynomial-time computable arguments.) So we have

$$\begin{aligned} PV_1 + \forall v\, \forall v'{<}D(u)\ \to \neg\psi(u,v,v') \vdash\ & \forall y_1, \ldots, y_k < D''(x) \\ & \neg\varphi(t_1(u), y_1, s_1) \vee \cdots \vee \neg\varphi(t_k(u, y_1, \ldots, y_{k-1}), y_k, s_k). \end{aligned}$$

By the deduction theorem, return the implication to its original direction to see what is needed:

$$\begin{aligned} & PV_1 \vdash \\ & \exists y_1, \ldots, y_k < D''(x)\ \big(\psi(t_1(u), y_1, s_1) \wedge \cdots \wedge \psi(t_k(u, y_1, \ldots, y_{k-1}), y_k, s_k)\big) \\ & \qquad\qquad \to \exists v\, \exists v'{<}D(u)\ \varphi(u,v,v'). \end{aligned}$$

$\dashv$

9.2 Applications

The dagger Herbrandization is essentially a harder alternative to the star Herbrandization. Its advantage is its applicability under weaker assumptions — the axiomatization suffices instead of the characterization. In the most important cases it turns out that it eventually provides the same links between Theorem 7.2, Theorem 5.2 and Theorem 6.1, as it was already shown in Section 8.2. These results are now presented mainly to illustrate the more general method, and to persuade the reader that the stronger notion of "characterization" indeed simplifies the constructions considerably.

Corollary 9.3 *The well-definedness of the class of $\Box_1^p$ functions taken as search problems axiomatizes the strictΣ_1^b-consequences of S_2^1.*

Proof: Assume $S_2^1 \vdash \forall u\, \exists v\, \theta(u,v)$, where θ is a *strict*Σ_1^b-formula. By Theorem 6.7, $\#FM$ captures the class of search problems uniformly defined by θ. By Theorem 9.2, $\#FM^\dagger$ captures the same class, too. To conclude the proof, it suffices to solve each $\#FM^\dagger$ search problem in polynomial time, provably in PV_1.

The instance of $\#FM^\dagger$ is:

$$(*) \qquad \forall x\, \exists y_1\, , \ldots , \exists y_k \bigwedge_{i=1}^{k} y_i \le t_i \wedge \Big(s_i \le t_i \rightarrow |f(t_i, y_i)| \le |f(t_i, s_i)| \Big)$$

where each t_i depends only on x and $y_1, \ldots, y_{i-1}$, whereas each s_i may depend on x and all $y_1, \ldots, y_k$.

Fix x and consider the following procedure. Take any $y = (y_1, \ldots, y_k)$, which satisfies $y_i < t_i$ for any i (the simplest choice is a k-tuple of zeroes). If $(*)$ holds (which is now a polynomial-time computable predicate), output y as a solution to this instance of $\#FM^\dagger$ and halt. Otherwise continue by taking the smallest i_l which falsifies the conjunction in $(*)$, and define a modified $y' = (y'_1, \ldots, y'_k)$ in this way:

$$y'_i = \begin{cases} y_i & \text{for } 1 \le i < i_l, \\ s_i & \text{for } i = i_l, \\ 0 & \text{for } i_l < i \le k. \end{cases}$$

Replace y with y' and repeat the procedure ad infinitum.

Argue in PV_1.

Observe that each y examined during the procedure is valid in this sense:

$$\forall i \; y_i < t_i(u, y_1, \ldots, y_{i-1})$$

by induction over the iterations.

Let $i = 1$. Observe that $f(t_i, y_i)$ changes only polynomially many times during the procedure. Continue by (metamathematical) induction over $i \leq k$ to see that the procedure terminates in a time bounded by a polynomial. $\dashv$

The fact that we have quietly provided a strong reduction in the sense of [BCE+95] is not surprising, because the reduction functions can always be easily absorbed into a target $\Box_1^p$ function taken as a search problem.

Corollary 9.4 *The well-definedness of polynomial local search problems axiomatizes the strictΣ_1^b-consequences of S_2^2.*

Proof: $FM^\dagger$ captures *strict*$\Sigma_1^b(S_2^2)$ by Theorem 5.11 and Theorem 9.2.

The instance of $FM^\dagger$ is:

$$(*) \qquad \forall x \; \exists y_1 \, , \ldots, \exists y_k \bigwedge_{i=1}^{k} y_i \leq t_i \wedge (s_i \leq t_i \rightarrow f(t_i, y_i) \leq f(t_i, s_i))$$

where each t_i depends only on x and $y_1, \ldots, y_{i-1}$, whereas each s_i may depend on x and all $y_1, \ldots, y_k$. Functions $N(x, y)$ and $v(x, y)$ are defined which impose the form of PLS on this search problem as follows.

Fix x and take any $y = (y_1, \ldots, y_k)$. If y satisfies $y_i < t_i$ for any i, define

$$v(x, y) = \sum_{i=1}^{k} f(t_i(x, y_1, \ldots, y_{i-1}), y_i) \cdot B^{k-i}(x),$$

where $B(x)$ is a common upper bound on the values of $f(x, y)$, $y_i < t_i$. If y satisfies every $y_i < t_i$, but breaks the conjunction in $(*)$, take the smallest i_l which breaks it, and define:

$$(N(x, y))_i = \begin{cases} y_i & \text{for } 1 \leq i < i_l, \\ s_i & \text{for } i = i_l, \\ 0 & \text{for } i_l < i \leq k. \end{cases}$$

and if y also satisfies the conjunction in $(*)$, define $N(x,y) = y$.

Finally, if some $y_i < t_i$ is broken or even y is not a k-tuple, define $N(x,y) = (0,\ldots,0)$ and $v(x,y) = v((0,\ldots,0)) + 1$.

It is straightforward to prove in PV_1 that this construction constitutes a many-one reduction of $FM^\dagger$ to PLS. $\dashv$

Another application of the dagger Herbrandization employs $GLS^\dagger$ for a new axiomatization result. It is perhaps useful to re-state the definition of $GLS^\dagger$ at this place, although it is still easiest understood in decomposition into Definition 7.3 and Definition 9.1.

Definition 9.5 *Let $k \geq 1$. The class of search problems $GLS^\dagger_k$ is defined by a polynomial-time computable predicate $a <^x_c b$, a polynomial-time computable function $v(x,a)$, polynomial-time computable functions $t_i(x,\ldots)$ of respective arities $4\cdot(i-1)+1$, for $1 \leq i \leq k$, and polynomial-time computable functions $s_i(x,\ldots)$ of arity $4k+1$, for $1 \leq i \leq k$.*

There is no promise.

The search task is to find a $4k$-tuple of elements

$$a_1, b_1, c_1, d_1, \ldots, a_k, b_k, c_k, d_k,$$

such that for every $1 \leq i \leq k$, all of

$$\begin{aligned} \neg s_i(x, a_1, \ldots, d_k) &<^{t_i(x,a_1,\ldots,d_{i-1})}_{a_i} b_i, \\ \neg s_i(x, a_1, \ldots, d_k) &<^{t_i(x,a_1,\ldots,d_{i-1})}_{b_i} c_i, \\ v(t_i(x, a_1, \ldots, d_{i-1}), b_i) &\leq v(t_i(x, a_1, \ldots, d_{i-1}), c_i), \end{aligned}$$

hold, or the transitivity of $<^{t_i}_{d_i}$ is broken by a_i, b_i, c_i like this:

$$a_i <^{t_i(x,a_1,\ldots,d_{i-1})}_{d_i} b_i \wedge b_i <^{t_i(x,a_1,\ldots,d_{i-1})}_{d_i} c_i \wedge \neg a_i <^{t_i(x,a_1,\ldots,d_{i-1})}_{d_i} c_i,$$

or the linearity or the antisymmetry of $<^{t_i}_{c_i}$ is broken in an analogous sense. The search problem $GLS^\dagger$ is the union of the $GLS^\dagger_k$ search problems for all k.

Theorem 9.6 *$GLS^\dagger$ axiomatizes the strictΣ^b_1-consequences of S^3_2.*

Proof: By Corollary 9.4, PLS axiomatizes *strict*$\Sigma^b_1(S^2_2)$ over PV_1. Through introducing new predicate symbols for Σ^b_1-definable predicates together with the defining axioms, that implies that $PLS^{\Sigma^p_1}$ axiomatizes *strict*$\Sigma^b_2(S^3_2)$ over S^2_2. By Lemma 7.4, GLS axiomatizes the same class over S^2_2. Because GLS is defined by a Π^b_1-formula, Lemma 2.12 shows that GLS axiomatizes the same class also over T^1_2.

Compose this with the fact that FM axiomatizes T^1_2 over PV_1 by (the proof of) Theorem 5.10 and the obvious fact that the GLS is a generalization of MIN which is itself a generalization of FM. ⊣

Note that an axiomatization of this class was previously given in [Fer95], under the name M_2, although it is not presented there as a search problem. Ferreira's M_2 (adapted to a search problem) and $GLS^\dagger$ look rather dissimilar, although any two search problems which axiomatize the same fraction must clearly be equivalent over PV_1.

Our methods fail to give immediate *uniformly* defined axiomatizations. But the results of [KP90], which we already discussed in Section 4.3.3, show that every fraction under consideration has a uniformly definable axiomatization. As a consequence, some $GLS^\dagger_k$ for a fixed k axiomatizes $\Sigma^b_1(S^3_2)$, too. A similar corollary applies to the result of Ferreira.

Appendix A

Three Myths About Function Problems

Current structural computational complexity theory puts its focus on the study of interesting classes of decision problems, also known as languages. Such problems are sets of strings and to solve them means to present a machine which gives a single bit of output for any candidate member of the set, thus computing its characteristic function. Part of the intuition of what constitutes an "interesting" class (or, less formally, an interesting computational model) is drawn from real world applications such as computer programming, as long as the intuition has a concise and robust mathematical expression.

On the other hand, genuine decision problems are somewhat rare in real world automated computations. Computer programmers do not live within, say, P or BPP, but, typically, within FP or $FBPP$, the corresponding classes of *function* problems. Each canonical description of an algorithm thus represents a partial function and problems are even partial multifunctions — there may be multiple fully acceptable outputs for the same input. No wonder that structural computational complexity theorists always preferred to concentrate on the mathematically more elegant decision problems. But the resulting discrepancy between the theory and the applications requires some justification.

It is the purpose of this Appendix to demonstrate that the common justification for replacing non-total function problems with the corresponding decision problems is philosophically inadequate. The Appendix is self-contained and independent even in the formal apparatus, because it needs to address a

more general topic. It is to be read as an optional appetizer before the thesis itself.

Problems, Algorithms, Reductions

Let us pick a formal basis for various computational models: Turing machines adapted to compute function problems. The machine has an input tape, working tapes, perhaps oracle query tapes or advice tapes, and an output tape. The content of the output tape at the moment of entering any accepting state is the *output* of the computation. In addition, a computation may end by entering a rejecting state, in which case the computation has no output. For accepting computations, the relation between the initial content of the input tape x and the final content of the output tape y is denoted as $M(x) = y$.

A very important detail is that rejections cannot act as inputs. Therefore the composed machine $f(g(x))$, where f and g are machines of the same kind, rejects whenever $g(x)$ rejects, as well as when $g(x) = y$ and $f(y)$ rejects.

Oracles, if present, can either solve decision or function problems. For decision problems, the answer can be indicated by oracle specific answer machine states; for function problems the oracle query tape also instantaneously changes its content from the input to any output consistent with the solved problem whenever the machine enters the positive oracle answer state.

For clarity, all reducibilities presented here will be based on different kinds of reductions performed in polynomial running time by a deterministic Turing machine, as this is the most popular combination of resource, bound and computational model. But the reasoning seems to be mostly independent of all three, as long as the general modus of many-one reductions and Turing reductions can be identified.

Definition A1 *A* function problem *is a partial multifunction $R(x, y)$. An algorithm* solving *this function problem within a computational model is any machine M based on that model, such that $M(x) = y \rightarrow R(x, y)$ and $M(x)$ rejects only if $\forall y \neg R(x, y)$.*

Definition A2 *A function problem $F(x, y)$ is (polynomial-time)* Turing reducible *to a function problem $G(x, y)$ (in symbols, $F \preceq_T G$) iff there is an oracle polynomial-time function which solves F whenever the oracle solves G.*

Further, a function problem $F(x, y)$ is f-bounded (polynomial-time) Turing reducible to a function problem $G(x, y)$ iff F is Turing reducible to G so that the number of oracle queries in any reduction computation is at most $f(n)$ where n is the size of x.

Interesting choices of f include 1, $O(1)$, $O(\log n)$, $O(\log^{O(1)} n)$ or $O(n)$.

The 1-bounded Turing reducibility should not be confused with many-one reducibility:

Definition A3 *A function problem $F(x, y)$ is (polynomial-time)* many-one reducible *to a function problem $G(x, y)$ (in symbols, $F \preceq_m G$) iff there are polynomial-time functions $f(x)$ and $g(x)$ such that*

$$F(x, g(G(f(x)))).$$

Myth One: Reducibility to Decision

Myth 1 *Important function problems are often self-reducible, and are thus equivalent to their decision versions.*

This is the key myth which in our opinion somewhat hinders the structural complexity research of function problems in contrast with the fast progress in the area of decision problems.

Discrepancy: Self-reducibility is indeed an important phenomenon[1], but there is an essential discrepancy between the types of reducibility here: whereas the standard reducibility in complexity research is the many-one reducibility, self-reducibility gives only a Turing reduction.

This gap cannot be overcome even in the textbook examples of SAT, graph 3-coloring etc. exactly because decision problems produce a smaller amount of information than (other) function problems. This is observed in the following fact.

[1]Giving e.g. $P = NP \iff FP = FNP$. This is, however, not an argument against an independent pursuit of the $FP \stackrel{?}{=} FNP$ question, but in favor of it. The method also seems to fall short of answering whether e.g. $RP = NP \iff FRP = FNP$; the structure of multifunctions and languages may be generally quite different.

Fact A1 *Take an NP-complete decision problem $D(x)$ and express it as $\exists y\, A(x,y)$ where $A(x,y)$ is computable in polynomial time, and where $A(x,y)$ implies $y > 0$ and the size of y is bounded by a polynomial in the size of x. Let $A'(x)$ be the function problem to output y such that $A(x,y)$ or zero when there is no such y.*

If $A'(x)$ is polylog-bounded Turing reducible to the decision problem $D(x)$, then $P = NP$.

Proof: Assume such a reduction R does exist. Consider the following decision algorithm F: given x, it simulates the computation of R, but instead of asking queries to $\exists y\, A(x,y)$ it tries out both possible answers in turn. As some of the branches are based on false assumptions, the assumed polylogarithmic bound on the depth of this recursion must be enforced during the simulation as well as the assumed polynomial bound on total running time of each branch. Every branch which does not exceed these limits outputs a candidate y; as the number of branches is polynomial and $A(x,y) \in P$, it is possible to verify the candidates. The algorithm F accepts if one of the candidate y's satisfies $A(x,y)$ and rejects otherwise.

The positive answer of F is always based on verifying a specific witness y and thus implies $\exists y\, A(x,y)$. On the other hand, suppose $\exists y\, A(x,y)$. Then the computation of $R(x)$ keeps within the bounds on running time and oracle queries and so its output was considered among the candidates found by F and so F will accept. F thus solves an NP-complete problem D in deterministic polynomial time. $\dashv$

Myth Two: Totality Makes Difference

Myth 2 *Total multifunctions are different from other function problems, having their corresponding decision problems constant and thus not being reducible to them.*

This view is so widespread that the only function problem classes that generally made it into introductory graduate courses on computational complexity until now are FP, FNP and $TFNP$, and merely $TFNP$ may hope that its popularity is not purely derived from its famous corresponding class of decision problems. It is, together with other total function problems, often introduced and motivated by this myth.

Discrepancy: The misunderstanding stems from a wrong choice of the corresponding decision problem for a function problem, which works in the case of self-reducible problems and fails in others, not necessarily total ones. But there is also a well-known strategy which works in *all* cases.

Fact A2 *Take $F(x)$ a function problem, whose solutions are of size bounded by a polynomial $p(|x|)$. Its corresponding decision problem is the following predicate $D(x, w, l)$:*

$$\exists y\ |y| < p(|x|) \wedge F(x) = y \wedge w = y_l$$

where y_l means "the first l bits of y".

Then F can be Turing-reduced to D.

Proof: Start by querying $D(x, 0, 0)$. If the answer is positive, a solution exists and it can be discovered bit by bit. In each step, it is already known that $D(x, w, l)$ for some w and l. Consequently

$$D(x, w, l+1) \vee D(x, 2^l + w, l+1),$$

and by asking a single query, it is possible to guarantee $D(x, w', l+1)$ for some w' in the next step. Accept in $p(|x|)$ steps with the last value of w. $\dashv$

Myth Three: Decisions Are Two-Valued

Myth 3 *Decision problems can be identified with two-valued function problems with regard to reductions.*

Discrepancy: The difference is that the set of possible outputs for a decision problem *including their semantics* is finite and fixed by the computational model, whereas with function problems this would be counterintuitive and in general impossible.

So the question whether $coSAT \preceq_m SAT$ is resolved in the positive when asked about two (total) function problems which just happen to have a two-valued range, whereas when asked about the decision problems, the answer is negative.

Anyway, it is possible to identify decision problems with partial *one-valued* function problems. In this way the unwanted output tape is disabled for decision problems.

Bibliography

[BCE+95] P. Beame, S. Cook, J. Edmonds, R. Impagliazzo, and T. Pitassi. The relative complexity of NP search problems. In *Proceedings of the 27th ACM Symposium on Theory of Computing*, pages 303–314, 1995.

[BDG88] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity.* Springer, 1988.

[BK94] S. R. Buss and J. Krajíček. An application of Boolean complexity to separation problems in bounded arithmetic. In *Proceedings of the London Mathematical Society*, volume 69, pages 1–27, 1994.

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

[Bus90] S. R. Buss. Axiomatizations and conservation results for fragments of bounded arithmetic. *Logic and Computation, Contemporary Mathematics*, 106:57–84, 1990.

[CK98] M. Chiari and J. Krajíček. Witnessing functions in bounded arithmetic and search problems. *Journal of Symbolic Logic*, 63(3):1095–1115, September 1998.

[CK99] M. Chiari and J. Krajíček. Lifting independence results in bounded arithmetic. *Archive for Mathematical Logic*, 38:123–138, 1999.

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

[Coo75] S. A. Cook. Feasibly constructive proofs and the propositional calculus. In *Proceedings of the 7th Annual ACM Symposium on Theory of Computing*, pages 83–97, 1975.

[Fer95] F. Ferreira. What are the $\forall\Sigma_1^b$-consequences of T_2^1 and T_2^2? *Annals of Pure and Applied Logic*, 75:79–88, 1995.

[HP98] P. Hájek and P. Pudlák. *Metamathematics of First-Order Arithmetic.* Perspectives in Mathematical Logic. Springer, second edition, 1998.

[JPY88] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer System Science*, 37:79–100, 1988.

[KP90] J. Krajíček and P. Pudlák. Quantified propositional calculi and fragments of bounded arithmetic. *Zeitschrift für Mathematikal Logik und Grundlagen der Mathematik*, 36(1):29–46, 1990.

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

[Kra93] J. Krajíček. Fragments of bounded arithmetic and bounded query classes. *Transactions of the A. M. S.*, 338(2):587–598, 1993.

[Kra95] J. Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory.* Cambridge University Press, 1995.

[LLS75] R. E. Ladner, N. A. Lynch, and A. L. Selman. A comparison of polynomial time reducibilities. *Theor. Comp. Sci.*, 1:103–124, 1975.

[Nel86] E. Nelson. Predicative arithmetic. *Mathematical Notes*, page 189, 1986.

[Pap94] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley Publishing Company, 1994.

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

[PY88] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *20th Annual ACM Symposium on Theory of Computing*, pages 229–234, 1988.

[Res86] J.-P. Ressayre. A conservation result for system of bounded arithmetic. Unpublished preprint, 1986.

[Rii93] S. Riis. Making infinite structures finite in models of second order arithmetic. *Arithmetic, Proof Theory and Computational Complexity*, pages 289–319, 1993.

[WP87] A. J. Wilkie and J. B. Paris. On the scheme of induction for bounded arithmetic formulas. *Annals of Pure and Applied Logic*, pages 261–302, 1987.