A Note on Satisfying Truth-Value Assignments of Boolean Formulas

Zbigniew Stachniak *

Department of Computer Science, York University, Toronto, Canada

Abstract. In this paper we define a class of truth-value assignments, called bounded assignments, using a certain substitutional property. We show that every satisfiable Boolean formula has at least one bounded assignment. This allows us to show that satisfying truth-value assignments of formulas in USAT can be syntactically defined in the language of classical propositional logic. We also discuss a possible application of bounded truth-value assignments in local search and other methods for solving Boolean satisfiability problems.

Introduction

In 1986, Valiant and Vazirani published a paper entitled *NP is as easy as detecting unique solutions* [8]. The paper presents a randomized reduction from SAT to the set USAT of all Boolean formulas with a unique satisfying assignment. The paper became almost instantly the instigator of vigorous research on USAT in the context of structural complexity theory (e.g. the study of randomized reductions [2], polynomial-time hierarchy [7], or of the complexity of function inversion [9]).

Clearly, the studies of USAT do not benefit the computational complexity theory exclusively. The use of high-performance SAT-solving methods (SAT solvers) in applied computer science hinges upon encoding of probelms as instances of propositional satisfiability (SAT). Empirical evidence indicates that computationally hard regions of same distributions of random formulas (such as random 3-CNF or random hyper-CNF [5]) not only coincide with thresholds between satisfiability and unsatisfiability but also occur near the regions where the number of formulas from USAT is the largest. Figure 1 illustrates this correspondence for random hyper-CNF formulas (defined in the next section): the number of uniquely satisfiable hyper-CNFs reaches its maximum at the threshold between satisfiability and unsatisfiability. Similar experiments performed on random 3-CNFs of 20 variables and m clauses show, for instance, that for a random sample of 1000 3CNFs, the number of uniquely satisfiable formulas sharply increases from 0 (for $m = 70$) to 80 at the threshold between satisfiability and unsatisfiability (at $m = 91$), and, then, it is followed by a steep decrease, reaching 0 when $m = 135$. This empirical data seems to indicate that the studies of USAT and, in particular, of properties of unique satisfying assignments, may provide insights into the development of novel techniques for guiding the search for satisfying assignments in SAT solvers.

The main result of this paper provides some evidence that such properties of unique assignments can be formulated. We prove that unique truth-value assignments of formulas in USAT can be defined syntactically in the language of propositional logic. To this end, we first define a class of truth-value assignments, called bonded assignments. In Section 2 we show that every satisfiable formula α has at least one bounded assignment and that among these assignments there is one (denoted as h_α) that can be defined syntactically. Hence the syntactic characterization of the unique satisfying assignment of $\alpha \in$ USAT is given by the definition of h_α.

The class of bounded truth-value assignments deserves some attention for a number of reasons. The definition of a bounded assignment can be used to develop new search heuristics for clausal as well as non-clausal SAT solvers (see Section 3). Furthermore, the definition of h_α constraints the set of all satisfying assignments of α in the sense that it contains enough information to rule out

* Research supported by the grant from the Natural Sciences and Engineering Research Council of Canada.

some truth-value assignments as satisfying α (cf. Lemma 2).

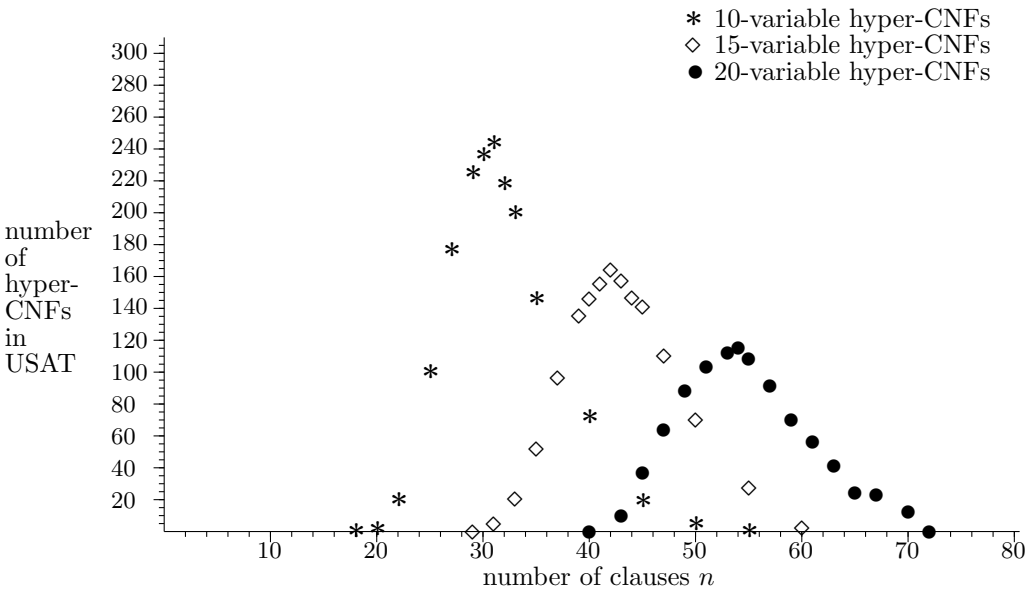

Figure 1. The number of uniquely satisfiable formulas from the class H-CNF(m, n) (hyper-CNFs) as a function of the number n of clauses. Sample size = 1,000 random hyper-CNFs, for every value n of clauses. The threshold between satisfiability and unsatisfiability for hyper-CNFs with 10 variables occurs when $n = 31$. For the hyper-CNFs with 15 and 20 variables, these thresholds are 42 and 54, respectively.

The remainder of this paper is structured as follows. Section 1 contains logical preliminaries. Section 2 sketches the theory of bounded truth-value assignments. In Section 3 we demonstrate a possible application of the theory by deriving a WalkSAT-like search heuristic. Finally, Section 4 contains the proofs of theoretical results reported in this paper.

1 Logical Preliminaries

The formulas of classical propositional logic are constructed, in the usual way, in terms of propositional variables, logical connectives (negation $\neg$, disjunction $\vee$, conjunction $\wedge$ and, possibly, other connectives) and the logical constant T (*truth*). A clause is a disjunction of literals (i.e., of variables and negated variables). A 3-CNF formula is a conjunction of clauses that contain exactly three literals each. A hyper-CNF formula is a Boolean formula of the form

$$(c_{11} \vee c_{12}) \wedge (c_{21} \vee c_{22}),$$

where c_{11}, c_{12}, c_{21}, and c_{22} are 3-CNF formulas of the same number of clauses. By H-CNF(m, n) we denote the class of all hyper-CNFs built in terms of m variables and such that every c_{11}, c_{12}, c_{21}, and c_{22} is a conjunction of exactly n clauses.

Example 1: The formula

$$(c_{11} \vee c_{12}) \wedge (c_{21} \vee c_{22}),$$

such that:

$$c11 = (p_0 \vee \neg p_1 \vee p_3) \wedge (p_0 \vee \neg p_2 \vee \neg p_3),$$
$$c12 = (p_1 \vee \neg p_2 \vee p_3) \wedge (p_0 \vee \neg p_1 \vee p_3),$$
$$c21 = (\neg p_0 \vee \neg p_1 \vee \neg p_3) \wedge (p_0 \vee \neg p_1 \vee \neg p_2),$$
$$c22 = (p_1 \vee p_2 \vee p_3) \wedge (p_0 \vee \neg p_1 \vee \neg p_3).$$

is a hyper-CNF from H-CNF$(4, 2)$. ■

A substitution is a mapping S that assigns a Boolean formula $S(p_i)$ to every variable p_i of some finite set $\{p_1, \ldots, p_n\}$ of propositional variables. We shall frequently represent a substitution S as a finite list of the form $[p_1/S(p_1), \ldots, p_n/S(p_n)]$ which explicitly indicates the assignments of formulas $S(p_i)$ to variables p_i, $1 \leq i \leq n$. If $S = [p_1/\alpha_1, \ldots, p_n/\alpha_n]$ is a substitution and $\alpha(p_1, \ldots, p_n)$ is a formula containing variables $p_1, \ldots, p_n$, then $S(\alpha)$–the application of S to α–is the formula obtained from α by the simultaneous replacement of every occurrence of every variable p_i with α_i. We shall frequently write $\alpha(p_1/\alpha_1, \ldots, p_n/\alpha_n)$ instead of $S(\alpha)$.

The notions of a truth-value assignment and of a satisfying truth-value assignment of a Boolean formula are defined in the usual way. We shall write $h(\beta) = 1$ to denote the fact that a formula β is true under a truth-value assignment h, and $h(\beta) = 0$ when β is false under h. Finally, if h is a truth-value assignment, p is a propositional variable, and $v \in \{0, 1\}$, then by $h[p/v]$ we denote the truth-value assignment defined exactly like h with the exception that $h[p/v](p) = v$.

2 Bounded Truth-Value Assignments

This section introduces the notion of a bounded truth-value assignment of a Boolean formula and reports basic facts concerning this class of assignments.

2.1 Bounded Truth-Value Assignments Defined

In his 1923 doctoral thesis entitled *On the Primitive Term of Logistic*, Alfred Tarski proved that the following formula is a theorem of the logical system called Prototheticc

Th. 66. $[f] : .\vartheta\rho\{f\} \supset : [\exists p].f(p). \equiv f(f(Vr))$

(cf, [6], Th. 66). From **Th. 66**, which Tarski calls the *second theorem on the upper bound of a function*, it follows that for every truth-function f, if $f(p)$ is true, then so is $f(f(Vr))$, where Vr is defined in Protothetic as the logical constant *verum*. Phrased in the language of propositional logic, **Th. 66** becomes the following metatheorem:

Lemma 1: *Let α be a Boolean formula containing a propositional variable p and let h be a truth-value assignment. If α is true under h, then so is the formula $\alpha(p/\alpha(p/T))$.*

Theorem **Th. 66** and Lemma 1 motivate the following definition.

Definition 1: *Let α be a Boolean formula. A truth-value assignment h is said to be bounded for α if:*

(b1) $h(\alpha) = 1$*;*
(b2) for every variable p that occurs in α, $h(p) = h(\alpha(p/T))$.

Example 2: The formula $\neg(p_1 \equiv p_2)$ has two satisfying truth-value assignments and both of them are bounded. On the other hand, among the satisfying assignments for $p_1 \rightarrow p_2$, only one, defined by $h(p_1) = h(p_2) = 1$, is bounded. ■

2.2 Syntactic Characterization of Bounded Assignments

While a satisfiable formula may have more than one bounded assignment, one of them can always be defined syntactically in the way described in Definitions 2 and 3 given below (see also Theorem 1).

Definition 2: *Let $\alpha(p_1, \ldots p_n)$ be a formula and let $p_1, \ldots, p_n$ be all the distinct variables that occur in α. The substitutions $S_1, \ldots, S_n$ are defined as follows.*

$S_1(p_1) = \alpha(p_1/T)$*;*

$S_1(q)$ is undefined, for every variable $q \neq p_1$.

Suppose that the substitution S_i has been defined, $i \geq 1$. Then

$S_{i+1}(p_{i+1}) = [p_{i+1}/T](S_i(\alpha))$;
$S_{i+1}(p_j) = [p_{i+1}/S_{i+1}(p_{i+1})](S_i(p_j))$, for every $1 \leq j \leq i$;
$S_{i+1}(q)$ is undefined, for every variable q not in $\{p_1, \ldots, p_{i+1}\}$.

EXAMPLE 3: Let $\alpha(p_1, p_2)$ be $\neg(p_1 \equiv p_2)$. Then:
$S_1(p_1) = \neg(T \equiv p_2)$;
$S_1(p_2) = undefined$;
$S_2(p_2) = [p_2/T](S_1(\neg(p_1 \equiv p_2))) = [p_2/T](\neg(\neg(T \equiv p_2) \equiv p_2)) = \neg(\neg(T \equiv T) \equiv T)$;
$S_2(p_1) = [p_2/S_2(p_2)](S_1(p_1)) = [p_2/S_2(p_2)](\neg(T \equiv p_2)) = \neg(T \equiv \neg(\neg(T \equiv T) \equiv T))$. ■

DEFINITION 3: *Let $\alpha(p_1, \ldots, p_n)$ and the substitutions $S_1, \ldots, S_n$ be as in Definition 2. Let h_α be the truth-value assignment defined as follows: for every variable $p_i, 1 \leq i \leq n$, if $S_n(p_i) \equiv T$, then $h_\alpha(p_i) = 1$; else $h_\alpha(p_i) = 0$ ($h_\alpha(q)$ is defined in an arbitrary manner for the remaining variables).*

EXAMPLE 4: Let $\alpha(p_1, p_2)$ be $\neg(p_1 \equiv p_2)$. Then, in view of Example 3,

$$S_2(p_1) \equiv \neg T \text{ and } S_2(p_2) \equiv T.$$

By Definition 3, $h_\alpha(p_1) = 0$ while $h_\alpha(p_2) = 1$. ■

The assignment h_α is defined syntactically in the language of propositional logic: for every p_i, $S_n(p_i)$ is a variable-free formula and, hence, it is equivalent to T or $\neg T$.

The following lemma describes the way h_α constarins the set of satisfying truth-value assignments of α.

LEMMA 2: *Let $\alpha(p_1, \ldots, p_n)$ be a satisfiable formula and suppose that for some $i \leq n$, $h_\alpha(p_i) = 0$. Then for every choice $v_1, \ldots, v_{i-1}$ of truth-values*

(a) $h_\alpha[p_1/v_1, \ldots, p_{i-1}/v_{i-1}, p_i/1](\alpha) = 0$.

In view of Lemma 2, the definition of h_α contains enough information to determine that some other truth-value assignments must falsify α.

THEOREM 1: *For every satisfiable Boolean formula α, h_α is a bounded truth-value assignment for α.*

Theorem 1 is the main theoretical result on bounded truth-value assignments reported in this paper. When restricted to USAT, it gives us:

COROLLARY 1: *If $\alpha \in$ USAT, then the only satisfying assignment of α is h_α.*

In view of Corollary 1 and Definition 3, the unique satisfying assignments of formulas in USAT are bounded and can be defined syntactically in the language of propositional logic. Definition 1(b2) provides a characterization of unique truth-value assignments of formulas in USAT.

3 Search for Bounded Truth-Value Assignment

Theoretical results presented in the previous section open the possibility of using properties of bounded truth-value assignments to derive new heuristics for guiding the search for satisfying truth-value assignments. In this section we test this hypothesis by showing how one can derive the search heuristic employed in the WalkSAT algorithm [3] from Definition 1.

A stochastic local search SAT solver is an incomplete procedure that performs a local search over the space of truth-value assignments (cf. Figure 2).

```
procedure xSAT(S)
    for i := 1 to MaxTries do
        h :=randomly chosen truth-value assignment
        for j := 1 to MaxFlips do
            if h(S) = {1} then return h
            C :=randomly selected clause
                such that h(C) = 0
            l(q) := select_literal(C, h)
            h(q) := 1 − h(q)
        end for
    end for
    return "satisfying assignment for S not found"
```

Figure 2. Generic stochastic local search algorithm for SAT.

Given an input CNF formula S, such an algorithm starts by generating a random truth-assignment h restricted to the variables that occur in clauses of S. Then, it locally modifies h until either a satisfying truth-value assignment is found or the limit on such modifications has been reached and the algorithm declares its failure in finding a satisfying assignment. A modification of a truth-value assignment h is done by, first, randomly selecting a clause C from S that is false under h and, then, by selecting a literal $l(q)$ of C and changing ('flipping') the truth-value of the variable q of $l(q)$ as to satisfy C. The literal $l(q)$ is selected using some search heuristic $select\_literal(C, h)$.

The condition (b2) of Definition 1 can be exploited in a number of ways to derive search heuristics. One can pursue an idea of minimization of the number of variables that violate (b2). Or one can try to develop a measure that indicates 'how close' a given variable is to satisfying (b2). In what follows, we shall pursue the later direction.

Let us consider a set S of clauses, a truth-value assignment h, a clause $C \in S$ that is false under h, and a literal $l(q)$ of C. Let $h^* = h[q/1 - h(q)]$ (i.e., h^* is obtained from h by flipping the truth-value of q). If $l(q) = q$, then $h(q) = 0$ and $h^*(q) = 1$. In order for q and h^* to satisfy (b2), $h^*(C'(q/T)) = h^*(C')$ should equal 1 for every clause C' that contains q. If this is the case, q is a good candidate for the selection. Otherwise, a natural course of action would be to select a variable that, after the flip of its truth-value, is the 'closest' to satisfying (b2). In other words, we should

(BH) select a variable p as to minimize the number of clauses that contain p and which are false when the truth-value of p is flipped.

Now, suppose that $l(q)$ is a negative literal (i.e, $l(q) = \neg q$). In this case $h(q) = 1$ and $h^*(q) = 0$. Since $h^*(C(q/T)) = h(C) = 0$, we conclude that q and h^* satisfy (b2). Of course, flipping any other negative literal of C will have the same effect. We can therefore use (BH) for breaking ties between negative literals. To conclude, we can use (BH) as a variable selection heuristic.

The heuristic (BH) is not entirely new. When combined with a random move, it results in the search heuristic of the WalkSAT algorithm (see [1], heuristic SKC). Nevertheless, the derivation of (BH) from (b2) can be viewed as an attempt at expanding the theory of local search by allowing the development of search heuristics based on properties of truth-value assignments. Non-clausal analogues of (BH) can also be derived from (b2) and applied in non-clausal local search SAT solvers such as polSAT (cf. [4,5]).

4 Technical Results

In this section we provide the proofs of Lemmas 1 and 2 and Theorems 1 and 2.

Proof of Lemma 1: Let α, p, and h be as stated and let us assume that $h(\alpha) = 1$.

If $h(\alpha(p/T)) = 1$, then $h(\alpha(p/\alpha(p/T))) = h(\alpha(p/T)) = 1$. If $h(\alpha(p/T)) = 0$, then $h(p) = 0$. Indeed, if $h(p) = 1$, then $h(\alpha) = h(\alpha(p/T)) = 1$ contradicting our assumption. So, $h(p) = 0 = h(\alpha(p/T))$ which gives us $h(\alpha(p/\alpha(p/T))) = h(\alpha(p)) = 1$. ■

Proof of Lemma 2: Let α and p_i be as stated. The condition (a) of Lemma 2 is equivalent to the following statement: for every sequence $v_1, \ldots, v_{i-1}$ of logical constants T and F (F is $\neg T$),

(a') $\alpha(p_1/v_1, \ldots, p_{i-1}/v_{i-1}, p_i/T, p_{i+1}/S_n(p_{i+1}), \ldots, p_n/S_n(p_n)) \equiv F$.

By the definition of S_n, $S_n(p_i)$ can be obtained from $\alpha(p_1/T, \ldots, p_i/T, p_{i+1}/S_n(p_{i+1}), \ldots, p_n/S_n(p_n))$ by a finite number of applications of the rewrite rule

$$T \Rightarrow \alpha(p_1/T, \ldots, p_i/T, p_{i+1}/S_n(p_{i+1}), \ldots, p_n/S_n(p_n))$$

that can be read as "replace an occurrence of T with $\alpha(p_1/T, \ldots, p_i/T, p_{i+1}/S_n(p_{i+1}), \ldots, p_n/S_n(p_n))$." Hence, $\alpha(p_1/T, \ldots, p_i/T, p_{i+1}/S_n(p_{i+1}), \ldots, p_n/S_n(p_n)) \equiv F$ (otherwise, $S_n(p_i) \equiv T$).

Now, suppose that there is a sequence $v_1, \ldots, v_{i-1}$ of logical constants T and F such that $\alpha(p_1/v_1, \ldots, p_{i-1}/v_{i-1}, p_i/T, p_{i+1}/S_n(p_{i+1}), \ldots, p_n/S_n(p_n)) \equiv T$. Then, in view of the definition of S_n, $S_n(p_i)$ can be obtained from $\alpha(p_1/v_1, \ldots, p_{i-1}/v_{i-1}, p_i/T, p_{i+1}/S_n(p_{i+1}), \ldots, p_n/S_n(p_n))$ by a finite number of applications of the rewrite rules:

$$F \Rightarrow \alpha(p_1/T, \ldots, p_i/T, p_{i+1}/S_n(p_{i+1}), \ldots, p_n/S_n(p_n)),$$
$$T \Rightarrow \alpha(p_1/v_1, \ldots, p_{i-1}/v_{i-1}, p_i/T, p_{i+1}/S_n(p_{i+1}), \ldots, p_n/S_n(p_n)).$$

This would mean that $S_n(p_i) \equiv T$ which contradicts our assumption. This concludes the proof of (a'). ■

LEMMA 3: *If α is a satisfiable Boolean formula, then $h_\alpha(\alpha) = 1$.*

Proof: Let h be any truth-value assignment that satisfies $\alpha(p_1, \ldots, p_n)$. First we show that for every $1 \leq i \leq n$,

(a) $h(S_i(\alpha)) = 1$.

From (a) we deduce the conclusion of the lemma in the following way. Since $S_n(\alpha)$ is variable free, $h(S_n(\alpha)) = h_\alpha(S_n(\alpha)) = h_\alpha(\alpha)$. So, in view of (a), we must have $h_\alpha(\alpha) = 1$.

To show (a), let us note that, by Lemma 1 and the fact that $h(\alpha) = 1$, we have $h(S_1(\alpha)) = h(\alpha(p_1/\alpha(p_1/T))) = 1$, as required.

Now, suppose that $h(S_i(\alpha)) = 1$, for some $i \geq 1$. By applying Lemma 1 to $S_i(\alpha)$ and p_{i+1} we get:

(b) $h(S_i(\alpha)(p_{i+1}/S_i(\alpha)(p_{i+1}/T))) = 1$.

But $S_{i+1}(\alpha) = S_i(\alpha)(p_{i+1}/S_{i+1}(p_{i+1}))$ and $S_{i+1}(p_{i+1}) = S_i(\alpha)(p_{i+1}/T)$. So, by (b), $h(S_{i+1}(\alpha)) = h(S_i(\alpha)(p_{i+1}/S_i(\alpha)(p_{i+1}/T))) = 1$, as required. ■

Proof of Theorem 1: Let $\alpha(p_1, \ldots p_n)$ be a satisfiable Boolean formula. By Lemma 3, Definition 1(b1) is satisfied.

To show that h_α also satisfies the condition (b2) of Definition 1, let p_i be such that $h_\alpha(p_i) = 1$. By Lemma 3, $h_\alpha(\alpha) = 1$. So, $h_\alpha(p_i) = 1 = h_\alpha(\alpha) = h_\alpha[p_i/1](\alpha) = h_\alpha(\alpha(p_i/T))$, as required.

Finally, suppose that for some $i \leq n, h_\alpha(p_i) = 0$. For every $1 \leq j \leq n$, let $v_j = h_\alpha(p_j)$. By Lemma 2, $h_\alpha(p_i) = 0 = h_\alpha[p_1/v_1, \ldots, p_{i-1}/v_{i-1}, p_i/1](\alpha) = h_\alpha(\alpha(p_i/T))$, as required. ■

5 Conclusions

In this paper we introduced the class of bounded truth-value assignments and showed that every $\alpha \in SAT$ has at least one bounded assignment. We proved that unique assignments of formulas in USAT are bounded and that they can be defined syntactically in the language of propositional logic. An interesting direction for future research is to further explore the properties of these assignments for the purpose of expanding the theory of local search and other methods for solving Boolean satisfiability problems.

References

[1] D. McAllester, B. Selman, and H. Kautz. Evidence for Invariants in Local Search. In Proc. of AAAI-97 (1997).

[2] R. Chang, J. Kadin, and P. Rohatgi. On Unique Satisfiability and the Threshold Behavior of Randomized Reductions. *Journal of Computer and System Sciences*, 50(3) (1995), pp. 359–373.

[3] B. Selman, H. Kautz, and B. Cohen. Noise Strategies for Local Search. In *Proc. of AAAI-94* (1994), pp. 337–343.

[4] Z. Stachniak. Polarity-based Stochastic Search Algorithms for Non-clausal Satisfiability. In *Beyond Two: Theory and Applications of Many-Valued Logic*, M. Fitting and E. Orłowska (eds.), Studies in Fuzziness and Soft Computing, Phisica-Verlag (2003), pp. 181–192.

[5] Z. Stachniak. Going Non-Clausal. In *Fifth International Symposium on the Theory and Applications of Satisfiability Testing, SAT 2002*, (2002), pp. 316–322.

[6] A. Tajtelbaum-Tarski. On the Primitive Term of Logistic. In *Leśniewski's Systems: Protothetic*, Z. Stachniak and J. Srzednicki (eds), Kluwer (1998), pp. 43–68.

[7] S. Toda. PP is as Hard as the Polynomial-Time Hierarchy. *SIAM Journal of Computing*, 20(5) (1991), pp. 865–877.

[8] L. G. Valiant and V.V. Vazirani. NP is as Easy as Detecting Unique Solutions. *Theoretical Computer Science*, 47 (1986), pp. 85 - 93.

[9] S. Watanabe and S. Toda. Structural analysis of the complexity of inverse functions. *Mathematical Systems Theory*, 26 (1993), pp. 203–214.