Simplifying the Weft Hierarchy

Jonathan F. Buss and Tarique Islam*

School of Computer Science, University of Waterloo
{jfbuss,tmislam}@uwaterloo.ca

Abstract. We give simple, self-contained proofs of the basic hardness results for the classes $W[t]$ of the weft hierarchy. We extend these proofs to higher levels of the hierarchy and illuminate the distinctions among its classes. The anti-monotone collapse at $W[1,s]$ and the normalization of weft-t formulas arise as by-products of the proofs.

1 Introduction

The theory of fixed-parameter hardness and the W-hierarchy has been obscured by technicalities of the definitions of the classes of the W-hierarchy and by the difficulty of the known proofs of the basic completeness results. We present a significantly simpler formulation and proofs of basic completeness results for $W[1]$, which eliminate many of the technicalities. The new proofs closely follow the development of the theory of *NP*-completeness, greatly simplifying the complex manipulations of circuits and eliminating the ancillary graph problems of the original proofs.

Starting from Chen and Flum's [3] characterization of $W[t]$, we present self-contained and relatively straightforward proofs that

1. WEIGHTED c-SAT $\in W[1]$ for all constants $c \geq 2$,
2. SHORT NTM ACCEPTANCE is hard for $W[1]$, and
3. SHORT NTM ACCEPTANCE is reducible to WEIGHTED 2-SAT via a reduction that produces only anti-monotone formulas.

The first proof uses a new algorithm for the Chen-Flum machine model, while the second and third were previously known. Using these proofs, an initial treatment of $W[1]$-completeness can follow entirely classical lines, closely mimicking the standard proofs that SAT is *NP*-complete:

1. SAT $\in$ *NP*,
2. A Turing machine can simulate a log-cost random-access machine in polynomial time, and
3. CNFSAT is *NP*-hard.

As an additional benefit of our proofs, we obtain the restriction to anti-monotone 2-SAT formulas as a simple observation.

* Both authors supported in part by a grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada.

The full impact of the new formulation arises higher in the W-hierarchy. We present new proofs that

1. WEIGHTED WEFT-t-DEPTH-d SAT $\in W[t]$ for all constants $d \geq t \geq 2$, and
2. WEIGHTED t-NORMALIZED SAT is hard for $W[t]$.

Our new proofs, unlike the former ones, are straightforward extensions of the $W[1]$ case. Moreover, the full Normalization Theorem arises as a simple corollary of the proofs.

Finally, we abstract the Chen-Flum characterization to a more general setting, comparable to but different than the "guess-then-check" classes of Cai and J. Chen [1]. The new viewpoint illuminates the apparent disconnection between alternation and space in parametric complexity [4].

2 Background

2.1 Parametric problems and the weft of circuits

Fellows, Downey and co-researchers (cf. [5]) developed the theory of fixed-parameter tractability based on concepts of "parameterized" problems and reductions. We present these notions briefly; for further discussion and details, see Downey and Fellows [5].

Definition 1. *A* parametric problem *is a set of pairs $Q \subseteq \Sigma^* \times \Gamma^*$ over finite alphabets Σ and Γ. For each $(x, k) \in Q$, x is taken as input and k is taken as the parameter. A parametric problem is* fixed-parameter tractable *if there is an algorithm that can determine whether $(x, k) \in Q$ using time $f(k)p(|x|)$ for some computable function f and polynomial p.*

(The parameter is often—in this paper, always—taken to be a positive integer. One may either code the integer over a suitable alphabet or simply use $\mathbb{N}$ in place of Γ^*.)

Definition 2. *An FPT*-reduction *from a parametric problem $Q \subseteq \Sigma^* \times \Gamma^*$ to a parametric problem $Q' \subseteq (\Sigma')^* \times (\Gamma')^*$ is a mapping $R\colon \Sigma^* \times \Gamma^* \mapsto (\Sigma')^* \times (\Gamma')^*$ such that*

1. *For all $(x, k) \in \Sigma^* \times \Gamma^*$, $(x, k) \in Q \Leftrightarrow R(x, k) \in Q'$,*
2. *There exists a computable function $g\colon \mathbb{N} \mapsto \mathbb{N}$ such that for all $(x, k) \in \Sigma^* \times \Gamma^*$, where $R(x, k) = (x', k')$, we have $k' \leq g(k)$, and*
3. *There exist a computable function $f : \mathbb{N} \mapsto \mathbb{N}$ and a polynomial p such that R is computable in time $f(k) \cdot p(|x|)$.*

We denote the existence of an FPT reduction from a parametric problem Q to another parameteric problem Q' by $Q \leq^{fpt} Q'$.

The breakthrough in measuring parametric hardness depended on a measure of circuit (or formula) complexity called *weft*. Fix any constant $u \geq 2$; call a gate of a circuit "large" if it has fan-in more than u.

Definition 3. *Let C be a decision circuit. The* weft *of C is the maximum number of large gates on any path from the inputs to the output. A circuit is t*-normalized *iff the and- and or-gates alternate, with an and-gate at the output, and it has depth (and hence weft) at most t.*

Since we will deal only with circuits of bounded depth and polynomial size, circuits are equivalent to formulas (tree circuits) of the same depth and polynomial size. Note that a formula in u-CNF is a product of sums having weft 1 and depth 2. Any CNF formula is 2-normalized.

A central problem in the theory of parametric complexity is the weighted satisfiability problem. An assignment to the variables of a formula (or circuit) has weight k iff it sets k variables to true and the rest false.

WEIGHTED SAT
Instance: A Boolean formula φ.
Parameter: An integer k.
Question: Does φ have a weight-k satisfying assignment?
WEIGHTED c-SAT (for $c \geq 2$)
WEIGHTED SAT restricted to c-CNF formulas.
WEIGHTED WEFT-t-DEPTH-d SAT (for $d \geq t \geq 2$)
WEIGHTED SAT restricted to formulas of weft t and depth d.
WEIGHTED t-NORMALIZED SAT (for $t \geq 2$)
WEIGHTED SAT restricted to t-normalized formulas.

The basic classes of the weft hierarchy are (essentially) defined by

1. $W[1, c]$ is the set of problems $\leq^{fpt}$-reducible to WEIGHTED c-SAT, for $c \geq 2$.
2. $W[t, d]$ is the set of problems $\leq^{fpt}$-reducible to WEIGHTED WEFT-t-DEPTH-d SAT, for $d \geq t \geq 2$.
3. $W[1] = \bigcup_{c \geq 2} W[1, c]$.
4. $W[t] = \bigcup_{d \geq t} W[t, d]$ for $t \geq 2$.

Much of the complexity of the seminal formulation of the W-hierarchy lies in the proofs of the following two results.

Proposition 4.

1. WEIGHTED c-SAT $\leq^{fpt}$ WEIGHTED 2-SAT *for all constants* $c \geq 2$.
2. WEIGHTED WEFT-t-DEPTH-d SAT $\leq^{fpt}$ WEIGHTED t-NORMALIZED SAT, *for* $d \geq t \geq 2$.

Each of these intermediate results was obtained via a complicated combinatorial proof including an ancillary graph-theoretic problem. For the first case, the proof defined and used the (novel) RED/BLUE NONBLOCKER problem; for the second case, the classical DOMINATINGSET problem sufficed. The details of reductions from satisfiability to the graph problems were crucial to the proofs. Although the two properties seem related, their proofs were nevertheless quite distinct. Below, we obtain both results as direct corollaries of our proofs. Also, our proof for weft $t \geq 2$ is simply an extension of the proof for weft 1.

2.2 Machines and algorithms

Chen and Flum [3], following Flum and Grohe [6], gave a new characterization of $W[t]$.

Definition 5 (Y. Chen, Flum). *A* W-RAM *is an alternating random-access machine with two sets of registers: standard registers $r_0, r_1, r_2, \ldots$, and guess registers $g_0, g_1, g_2, \ldots$. Any standard deterministic operation is allowed on the standard registers. Only the following operations may access the guess registers:*

Exists j: Store any natural number in the range $[0, r_0]$ in register g_j, with existential branching.
Forall j: Store any natural number in the range $[0, r_0]$ in register g_j, with universal branching.
JG= i j c: If $g_{r_i} = g_{r_j}$, then jump to instruction location c.
JG0 i j c: If $r_{\langle g_{r_i}, g_{r_j} \rangle} = 0$, then jump to instruction location c.

Here $\langle \cdot, \cdot \rangle$ is any suitable pairing function.

The JG0 instruction is essentially a test of a binary relation, encoded in the standard registers.

Limiting the access to guess registers simplifies the analysis of a computation. The following lemma describes a crucial property of computations by W-RAMs. The model could be modified by allowing additional operations on the guess registers, so long as this property is preserved.

Lemma 6 (Y. Chen, Flum). *If two executions of a W-RAM on the same input follow the same sequence of instructions, they have the same set of values in the standard registers.*

Definition 7 (Y. Chen, Flum). *A W-RAM program R is an* AW-program *if there are a computable function f and a polynomial p such that for every input (x, k) with $|x| = n$ the program R on every run,*

1. *performs at most $f(k) \cdot p(n)$ steps,*
2. *has at most $f(k)$ existential or universal steps,*
3. *accesses at most the first $f(k) \cdot p(n)$ standard registers, and*
4. *stores values at most $f(k) \cdot p(n)$ in any register at any time.*

We shall treat the following result of Chen, Flum and Grohe [4] as the definition of the classes $W[t]$.

Proposition 8 (Y. Chen, Flum, Grohe). *Let Q be a parameterized problem and $t \geq 1$. Then Q is in $W[t]$ if and only if there are a positive integer u, a computable function h and an AW-program R for a W-RAM such that R decides Q and such that for every run of R on an instance (x, k) of Q as input,*

1. *all existential and universal steps are among the last $h(k)$ steps of the computation,*
2. *there are at most $t - 1$ alternations between existential and universal states, the first guess step being existential, and*
3. *every block without alternations, besides the first one, contains at most u guess steps.*[1]

Given this characterization, simple algorithms suffice to put some problems in the W-hierarchy.

[1] Omitting this condition gives apparently different classes denoted $A[t]$ [6]. With hindsight, one may consider that the difficulty of formulating this condition in terms of Turing machines or circuits lies at the heart of the difficulty in creating the original formulation of parameterized hardness.

Lemma 9.

– *The following are in* $W[1]$*:* INDEPENDENTSET, CLIQUE, RED/BLUE NONBLOCKER *and* SHORT NTM ACCEPTANCE.
– DOMINATINGSET *is in* $W[2]$.

Proof. For the first set of problems, the requisite property depends on a binary relation — the adjacency matrix for a graph or the successor relation for a Turing machine. An AW-program can write the relation into the first n^2 standard registers. It then existentially guesses k nodes (or configurations) and checks that they meet the requirement.

DOMINATINGSETis similar, except that the AW-program universally guesses an additional node and checks that it is dominated by one of the existentially guessed nodes.

For some other cases, the property of interest requires more than a single binary relation. An AW-algorithm must carefully code information for access during the checking phase. We illustrate the process with weighted-satisfiability problems in the next section.

3 Algorithms for Weighted-Satisfiability Problems

Lemma 10. WEIGHTED c-SAT *is in* $W[1]$*, for all constants* c.

Proof. The basic relation of interest has arity c: given a partial assignment of boolean values to c variables, does it falsify any clause? To make this relation sufficiently accessible to an AW-program, we use a modified view of it.

– We refer to negated variables in groups, rather than individually.
– We reinterpret the c-ary relation as a unary relation over c-tuples of variables.

Let C be a c-CNF circuit over variables $X = \{x_1, \ldots, x_n\}$. Introduce $n' = \binom{n+2}{2}$ "gap variables" $Y = \{\, y_{i\ell} \mid 0 \leq i \leq n$ and $0 \leq \ell \leq n - i \,\}$, where $y_{i\ell}$ is to be true exactly when all variables $x_{i+1}, \ldots, x_{i+\ell}$ are false ($\ell = 0$ gives the empty range, for the case of two true variables with consecutive indices).[2] Further, add $n'' = \binom{n+n'}{c}$ variables $Z = \{v_1, \ldots, v_{n''}\}$ over $[n + n']^c$, each specifying a size-c set of x and y variables. Index the union $X \cup Y \cup Z$ from 1 to $n + n' + n''$.

To guess a weight-k assignment, an AW-program will guess k indices of variables from X, $k+1$ indices of variables from Y, and $\binom{2k+1}{3}$ indices of variables from Z. To check consistency of the guesses and their satisfaction of the circuit, the program prepares a table A that has $A_{ij} = 1$ if one of the following holds and $A_{ij} = 0$ otherwise.

1. $i = 0$ and $1 \leq j - 3 \leq n$ (i.e., $j - 3$ codes a variable),
2. $i = 1$ and $n < j - 3 \leq n' + n$ (i.e., $j - 3$ codes a gap),
3. $i = 2$ and $n' + n < j - 3 \leq n'' + n' + n$ (i.e., $j - 3$ codes a triple) and $j - 3$ does not falsify any clause of the circuit, or
4. $i \geq 3$, $j \geq 3$, and one of
   (a) variable $i - 3$ occurs at the end of gap $j - 3$,

[2] We have borrowed the idea of gap variables from the original proof. They form the only remnant of the RED/BLUE NONBLOCKER and DOMINATINGSET problems.

(b) gap $i-3$ starts after variable $j-3$, or
(c) tuple $i-3$ contains variable or gap $j-3$.

The table A contains the necessary information to confirm that a guessed set of variables consistently satisfies the circuit. To access the first three rows of a table, the AW-program also guesses $g_0 \leq 0$, $g_1 \leq 1$, and $g_2 \leq 2$ and requires that all three be distinct.

The technique used above to combine different types of information into a single binary relation extends to other situations. Suppose that an AW-program needs information from d relations $Q_1, \ldots, Q_d$ of arities $c_1, \ldots, c_d$ over domains $D_1, \ldots, D_d$. The program uses codes for c_i-tuples from D_i, for each i, and sets up a single binary "master" relation as above. It then guesses the required domain variables and the codes of the corresponding tuples, and confirms the necessary relationships among them. The time to set up the "master" relation is at most $O\left(\sum_i |D_i|^{c_i}\right)$, and the time to check the guesses depends on d and the number of guesses. If d is constant, the domain sizes $|D_i|$ are parametric polynomial, and the number of guesses is parametric, the entire AW-program will meet the conditions for some level of the W-hierarchy.

Lemma 11. WEIGHTED WEFT-t-DEPTH-d SAT $\in W[t]$, *for* $d \geq t \geq 2$.

Proof. Fix any constant d, and let φ be a weft-t, depth-d tree circuit with negations only at the inputs. The algorithm has three main phases: simplifying the form of the circuit, constructing tables to enable fast truth-checking, and finally evaluating the circuit at guessed values using alternation.

1. Simplify the form of the circuit, to have (starting from the output) at most two levels of small gates, followed by t levels of alternating large gates (no small gates), and then one level of small gates. The algorithm proceeds from the output of the circuit to the inputs, locally restructuring the circuit so as to combine each small gate with a neighbouring large gate.[3]
2. Construct satisfaction-checking tables (as in the algorithm for CNFSAT above) for the large gates closest to the input literals.
3. Guess k variables to be true, find inputs to $t-1$ levels of large gates using alternation, and finally check the satisfaction of the input gates using the tables constructed in phase 2.

Simplifying the form of a circuit. Let a "small sub-circuit" be a maximal connected sub-circuit comprising only small gates. Since each small sub-circuit has gates of fan-in two and has depth at most d, it receives at most 2^d "input" values (each either a literal or the output of a large gate). Therefore, its CNF and DNF equivalents each have size at most 2^{2^d}, which is a constant for our purposes. We will select one of these equivalents depending on the surrounding circuit.

[3] This process is essentially the one used as a (small) part of the original proof of the Normalization Theorem ([5], p. 293). The tables of phase 2, combined with the alternation of phase 3, replace the multiple changes of variables and bypass the DOMINATINGSET problem entirely. Our following proof of the $W[t]$-completeness of WEIGHTED WEFT-t SAT will complete the normalization.

As a first step, convert the small sub-circuit (if any) at the output either to CNF or to DNF. Next consider a large gate G whose output is an input to this sub-circuit. We shall assume for simplicity of description that G is an and-gate. (If G is an or-gate, apply the De Morgan dual of the following.) Each input G_i of G comes from another small sub-circuit S_i. Convert each S_i to its CNF equivalent S_i', and merge the output gates of each S_i' into G, forming a larger and-gate G'.

Each or-gate S_{ij} of each S_i' has at most 2^d gates as inputs. Those that are or-gates may logically be merged with S_{ij}, but to do so would increase the weft if S_{ij} also has a large and-gate as input. To circumvent this problem, consider the partial circuit comprising gate S_{ij} and all of its large and-gate inputs. (The inputs to this partial circuit come from or-gates, small sub-circuits, and literals.) The partial circuit is in DNF form, with a constant number (2^d) of disjuncts. Therefore, its CNF equivalent circuit C_{ij} has size $n^{O(1)}$. Replace the partial circuit by C_{ij} and finally merge the output of C_{ij} with the original gate G.

To illustrate (see Fig. 1), suppose one input to a large and-gate G is $G_1 = (V \wedge W) \vee A$, where V and W are large or-gates and $A = \bigwedge_{j=1}^{m} A_j$ is a large and-gate. Converting the small gates to CNF yields $(V \vee A) \wedge (W \vee A)$. Expand $V \vee A$ into $\bigwedge_{j=1}^{m} (V \vee A_j)$ and similarly for W. The final circuit replacing G_1 is thus $\bigwedge_{j=1}^{m} (V \vee A_j) \wedge \bigwedge_{j=1}^{m} (W \vee A_j)$.

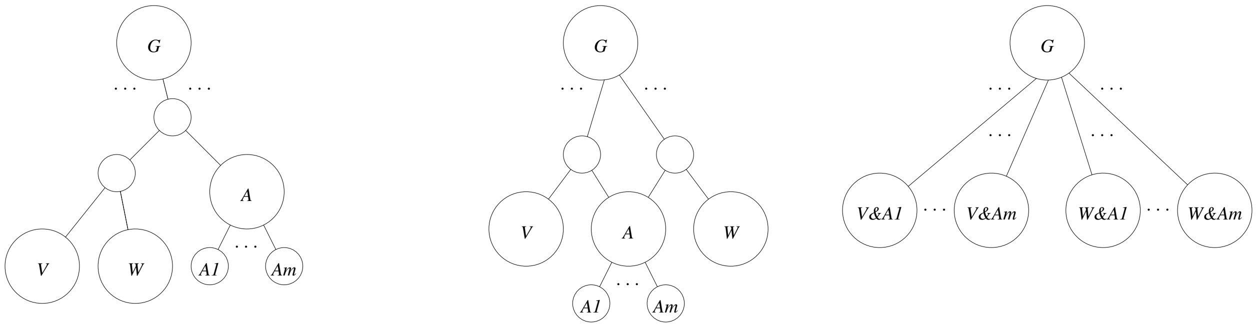

Fig. 1. The two steps of removing a small sub-circuit. G and A are large and-gates; V and W are large or-gates.

At this point the gate G (as modified) has only large gates and literals as inputs. Move to the next level of large gates, and convert them in the same manner. After the final step, when no unconverted large gates remain, there will be one layer of small gates at the inputs, giving the required form.

Creating the tables for satisfaction-checking. Following the simplification, each depth-2 circuit at the inputs, comprising a single large gate whose inputs are small gates with literals as inputs, is in either 2^d-CNF (for a large and-gate) or 2^d-CNF (for a large or-gate) form. Each of these depth-2 circuits receives its own tables for checking satisfaction, indexed by variable, gap and 2^d-tuple, as in Lemma 10.

Evaluating the circuit. To check satisfiability of the resulting weft-t circuit by a weight-k assignment, existentially guess both the assignment and the values of the small monotone sub-circuit at the output. Proceed toward the inputs, guessing an input line for each large gate (existentially for an or-gate and universally for an and-gate), until an input depth-2 circuit is reached. Finally, check that the required conditions hold: each large gate computes its output correctly, the output sub-circuit produces *true*, and the required output of the input circuit is justified by its corresponding satisfaction table.

4 Hardness of Weighted-Satisfiability Problems

4.1 Level $W[1]$

For the needed hardness results, we use previous work concerning Turing machines—specifically the SHORT NTM ACCEPTANCE problem. The following two lemmas provide the connections to satisfiability and to AW-programs (Cai, L. Chen, Downey, Fellows [2]).

Lemma 12. SHORT NTM ACCEPTANCE *is hard for* $W[1]$.

Lemma 13. SHORT NTM ACCEPTANCE $\leq^{fpt}$ WEIGHTED 2-AMSAT.

Together with the above algorithm for WEIGHTED c-SAT, these yield a new proof of the "Levin-Cook Theorem" for $W[1]$.

Theorem 14. SHORT NTM ACCEPTANCE, WEIGHTED 2-AMSAT *and* WEIGHTED c-SAT *(for any fixed* $c \geq 2$*) are complete for* $W[1]$.

Well-known reductions then give

Corollary 15. INDEPENDENTSET, CLIQUE, *and* RED/BLUE NONBLOCKER *are complete for* $W[1]$.

We reprise the original proofs of the lemmas (cf. [5]) in order to later extend them higher in the weft hierarchy.

Proof. (of Lemma 12). Let $Q \in W[1]$ be a parametric problem solved by AW-program R, using $f(k)p(|x|)$ steps and ending with $h(k)$ guess steps, on any input (x, k). (Hereinafter we shall omit the arguments of f, p, and h.) The desired reduction, on input (x, k), will produce a nondeterministic Turing machine M that accepts the empty input in k steps if and only if $(x, k) \in Q$. M will have alphabet $\Sigma = [f \cdot p]$; that is, one symbol for each possible value of a register of R. The states of M will encode $(h+1)$-tuples $\langle S, i_1 \ldots, i_h \rangle$, where S is an instruction of R and $i_1, \ldots, i_h$ are indices of the relevant registers.

The start state of M encodes the configuration of $R(x)$ reached just before the first guess step. M begins by guessing the h values for the existential registers of R, writing one guess to each of h cells. It then checks, in h^2 steps, that R would indeed accept with these guesses. Thus M accepts the empty input in $h^2(k)$ steps iff R accepts (x, k), as required.

Proof. (of Lemma 13). The proof modifies classical proofs of Cook's Theorem that SAT is NP-complete, with a view to creating clauses of the required form.

Let M have state-set Q and alphabet Σ (including blank). Consider the tableau associated with a computation of M. Since we are interested in the first k steps of any computation of M, the tableau will contain k rows and k columns, for a total of k^2 entries. Without loss of generality we assume that M has a unique halting state $q_{\rm yes}$ and that once M enters state $q_{\rm yes}$ it remains there indefinitely.

The contents of this tableau are described using variables $Y_{tph\tau\sigma}$, where

$$1 \le t \le k, \quad 1 \le p \le k, \quad h \le k, \quad \tau \in Q, \quad \text{and} \quad \sigma \in \Sigma \ .$$

The reduction produces a formula φ that is satisfied by a weight-k^2 assignment A of the variables if and only if there is a computation of M such that for all t, p, h, q and σ, variable $Y_{tmhq\sigma}$ has value *true* in A exactly if at time t, position p of the tape contains symbol σ and the head scans position h in state q.

This set of variables is chosen in order to obtain anti-monotone 2-CNF formulas. For the same reason, we do not directly require consistency among the parts of the computation; instead, we forbid inconsistency. All clauses will have the form $w \Rightarrow \neg x$, which we write as $\neg w \vee \neg x$. For example, the condition that at most one state occurs at time t is expressed by the conjunction

$$\prod_{1 \le p,p' \le k} \quad \prod_{1 \le h,h' \le k} \quad \prod_{\sigma,\sigma' \in \Sigma} \quad \prod_{q \ne q' \in Q} \quad (\neg Y_{tphq\sigma} \vee \neg Y_{tp'h'q'\sigma'}) \ .$$

The constraints on transitions are enforced similarly. For example, if the head is not at position p at time t, the symbol there cannot change at time $t+1$:

$$\prod_{1 \le h,h' \le k} \quad \prod_{q,q' \in Q} \quad \prod_{\substack{1 \le p \le k \\ p \ne h}} \quad \prod_{\sigma \ne \sigma' \in \Sigma} \quad \left(\neg Y_{tphq\sigma} \vee \neg Y_{(t+1)ph'q'\sigma'}\right) .$$

Similar conjunctions ensure that no incorrect state transition occurs, and thus that the transition that must occur is correct. We leave the conditions on the start configuration, the boundaries of the tableau, and ending in an accepting state at time k entirely to the reader.

The condition that some combination of state, head position and cell contents must hold at each step is not directly expressed by the formula. This condition is enforced by the consideration only of assignments of weight k^2 in combination with the expressed prohibitions.

The entire formula has size $f'(k)p'(n)$ and can be computed in proportional time. It has a satisfying assignment of weight k^2 if and only if R accepts the empty input, as required.

Looking ahead, we note that one can substitute monotone CNF formulas (with unbounded conjuncts) for the anti-monotone 2-CNF formulas of the proof. For example, the condition that at least one state occurs at time t is expressed by the formula

$$\sum_{1 \le p \le k} \quad \sum_{1 \le h \le k} \quad \sum_{\sigma \in \Sigma} \quad \sum_{q \in Q} \quad Y_{tphq\sigma} \ .$$

With all subformulas modified in this fashion, the restriction to assignments of weight k^2 ensures that multiple states do not occur.

4.2 Levels $W[t]$, for $t \geq 2$

To translate the above results to $W[t]$ for $t \geq 2$, we combine the proofs of Lemmas 12 and 13. The key additional technique is to handle all of the guess steps after the first existential block deterministically, as part of the reduction itself. In other words, for every configuration that could occur at the end of the first existential block, the reduction computes the conditions under which the machine would eventually accept from that configuration and incorporates these conditions into the produced formula.

Theorem 16. *For each $t \geq 2$, WEIGHTED t-NORMALIZED SAT is complete for $W[t]$.*

Corollary 17. DOMINATINGSET *is $W[2]$-complete.*

Proof. Let AW-program R accept Q in time $f(k)p(n)$, with guess steps only the last $h(k)$ steps and occurring in t alternating blocks, with existential first and all but the first limited to u guesses. We consider a computation by R to have three parts: (1) the deterministic part, up until the first guess step, (2) the existential part, comprising the steps from the first guess step up to the first universal step, and (3) the decision part, comprising the remainder of the computation. We handle the first two parts by combining the constructions of Lemmas 12 and 13, producing a set φ_E of clauses that mimic the computation by R. However, we omit the clauses that require the (virtual) Turing machine to end in an accepting state. We will replace the omitted clauses with a formula φ_A that requires part 3 of R's computation to be accepting, as given below. Formula φ_A will use the same variables as φ_E. (As noted above, φ can be taken in either monotone or anti-monotone form, as required.)

For simplicity, we start with the case $t = 2$; that is, the decision part comprises a single universal block with u guesses. For each possible outcome $G = \langle g_1, g_2, \ldots, g_u \rangle$ of the universal guesses and each possible branch b determined by outcomes of the tests involving the earlier existential guesses, the reduction will produce a formula $\alpha_{G,b}$ described below. (Recall Lemma 6: the values of standard registers do not depend on the precise values in the guess registers, but only on the branching they produce.) With any assignment to φ_E that represents a valid computation of the existential part, the formula $\alpha_{G,b}$ will evaluate to true if the universal choices specified by G lead to acceptance along path b or if some path other than b is followed, and will evaluate to false if the universal choices given by G lead to rejection along path b. Thus the conjunction of all formulas $\alpha_{G,b}$ gives the required acceptance condition for the computation of R. Since there are $(fp)^u$ possible outcomes of the u universal guesses and 2^h possible branches in h steps, the overall formula meets the required bounds on size.

If branch b ends by accepting, using $\alpha_{Gb} = \mathit{true}$ suffices. The case that b rejects requires a non-trivial formula, based on sub-formulas for each test instruction T on branch b that involves an existential register (the values of universal registers are specified by G). Each subformula will evaluate to true if test T is *not* satisfied. For a given test instruction T, we shall actually define four different formulas. The formulas π_T and σ_T will be satisfied if T is taken, and $\pi_{\neg T}$ and $\sigma_{\neg T}$ will be satisfied if T fails. The π formulas will be anti-monotone products-of-sums and the σ formulas monotone sums-of-products.

Let $Y_{j\ell}$ be the variable (full name: $Y_{k1jq_{yes}\ell}$) that is true iff existential-guess register j receives value ℓ during part 2 of R's computation. We define the subformulas σ_T and π_T as follows. Note that the correctness of these formulas depends on the fact that their satisfaction is only relevant when φ_E is satisfied, *i.e.*, when exactly one Y_{jl} is true for each value of j.

T is JG= $i\ j\ c$:

If i is universal and j existential, let

$$\sigma_T = Y_{r_j g_{r_i}} \qquad \text{and} \qquad \pi_T = \prod_{\ell \neq g_{r_i}} \neg Y_{r_j \ell} \ .$$

If both i and j are existential, let

$$\sigma_T = \sum_{0 \leq \ell \leq fp} Y_{r_i \ell} \wedge Y_{r_j \ell} \qquad \text{and} \qquad \pi_T = \prod_{0 \leq \ell \neq \ell' \leq fp} \neg Y_{r_i \ell} \vee \neg Y_{r_j \ell'} \ .$$

T is JG0 $i\ j\ c$:

If i is universal and j existential, let

$$\sigma_T = \sum_{\left\{ \ell \,|\, r_{\langle g_{r_i}, \ell \rangle} = 0 \right\}} Y_{r_j \ell} \qquad \text{and} \qquad \pi_T = \prod_{\left\{ \ell \,|\, r_{\langle g_{r_i}, \ell \rangle} \neq 0 \right\}} \neg Y_{r_j \ell} \ .$$

If both i and j are existential, let

$$\sigma_T = \sum_{\left\{ \ell, \ell' \,|\, r_{\langle \ell, \ell' \rangle} = 0 \right\}} Y_{r_i \ell} \wedge Y_{r_j \ell'} \qquad \text{and} \qquad \pi_T = \prod_{\left\{ \ell, \ell' \,|\, r_{\langle \ell, \ell' \rangle} \neq 0 \right\}} \neg Y_{r_i \ell} \vee \neg Y_{r_j \ell'} \ .$$

For any test T, the formulas $\sigma_{\neg T}$ and $\pi_{\neg T}$ are obtained as the normal-form equivalents of $\neg \pi_T$ and of $\neg \sigma_T$, respectively.

We can obtain the desired weft by taking

$$\alpha_{G,b} = \sum_{T \text{ in } b} \sigma_{\neg T}$$

and then

$$\varphi_A = \prod_G \prod_b \alpha_{G,b}$$

The formula has depth 3, however, due to the binary conjunctions in the σ formulas. To reduce the depth, we introduce new variables Z_{cd} for $0 \leq c, d \leq k^2 fp$, with the intention that $Z_{\langle i_1, j_1 \rangle \langle i_2, j_2 \rangle} \Leftrightarrow (Y_{i_1 j_1} \wedge Y_{i_2 j_2})$. In order to obtain a monotone formula, we rely on the weight restriction and express directly only half of the conditions on the Z variables, using the requirement

$$\varphi'_Z = \prod_{\substack{1 \leq i,j \leq k \\ 1 \leq \ell, m \leq fp}} Z_{\langle i,l \rangle \langle j,m \rangle} \vee \neg Y_{i\ell} \vee \neg Y_{jm}$$

in its equivalent form

$$\varphi_Z = \prod_{\substack{1\le i,j\le k\\ 1\le \ell,m\le fp}} \left(Z_{\langle i,\ell\rangle\langle j,m\rangle} \vee \sum_{\ell'\neq\ell'} Y_{i\ell'} \vee \sum_{m'\neq m} Y_{jm'} \right) \ .$$

If formula φ_Z is satisfied by $k^2 + \binom{k^2}{2}$ true variables, these must include exactly k^2 Y variables and $\binom{k^2}{2}$ Z variables.

Thus replace each occurrence of $Y_{i\ell} \wedge Y_{jm}$ in φ_A by $Z_{\langle i,\ell\rangle\langle j,m\rangle}$. With this substitution, the monotone CNF formula $\varphi_A \wedge \varphi_Z$ evaluates to true on an assignment of weight $k + \binom{k^2}{2}$ if and only if R accepts after the corresponding existential guesses. The size of the formula is $g(k)e(n)$ for a computable function g and a polynomial e (which depend on the constants t and u, as well as functions h and f).

The final formula $\varphi_E \wedge \varphi_A \wedge \varphi_Z$ has depth 2, has size polynomial in n and k, and has a satisfying assignment of weight $k + \binom{k^2}{2}$ if and only if AW-machine R accepts its input, as required. This completes the reduction for the case $t = 2$.

For higher values of t, the same idea applies, except that each α formula has depth $(t-1)$, giving an overall weft of t. The formulas can be taken as monotone (using the σ subformulas) for even values of t, and anti-monotone (using the π subformulas) for odd values of t. Likewise, in the formula φ_E representing parts 1 and 2 of the computation of R, one can use either the monotone version or the anti-monotone version, as desired.

We have therefore reduced problem $Q \in W[t]$ to WEIGHTED t-NORMALIZED SAT, as required.

5 Remarks

Our proofs of completeness for the classes of the W-hierarchy are conceptually much simpler than the originals. We anticipate that they will make the theory of fixed-parameter intractability more accessible and lead to further results.

Considering only the characterization of $W[t]$ by AW-programs, one may reasonably ask whether these classes really represent the "right" definition of hardness classes. The limited use of the guess registers and the restriction to a constant number of guesses in each block after the first seem rather arbitrary. To date, however, the classes $A[t]$ and $L[t]$ that arise without these restrictions [6, 3] do not seem to have natural compete problems of independent interest. In contrast, $W[2]$ (for example) has importance as the class of languages FPT-reducible to DOMINATINGSET.

If desired, the severe restrictions on use of guess registers in the W-RAM model can be relaxed. Appropriate relaxations will avoid some technicalities of the satisfiability algorithms presented here. So long as the relaxed model continues to satisfy Lemma 6, the hardness results will hold as presented here.

Cai and J. Chen [1] proposed a model of "guess then check" computations: their classes $GC(s(n), \mathcal{C})$ contain the problems that can be solved by existentially guessing a witness of $s(n)$ bits and then checking the witness within complexity class $\mathcal{C}$. The AW-programs considered here use a similar but more general paradigm that one might call "prepare, guess, and check." They have an preparatory deterministic part, followed by existential guesses and finally a check of correctness. For example, $W[1]$ corresponds to parameterized polynomial time followed by a parametric constant number of guesses (of $O(\log n)$ bits each) with a parametric constant number of checking operations (with operands of $O(\log n)$ bits). Thus "$PGC(FPT, NTIME(\log n))$" is a suitable characterization of the class. (Here we interpret "$\log n$" as the logarithm of the length of the output from the preparation phase; thus it includes any factors depending on the parameter.) For the classes $W[t]$ ($t \geq 2$), the checking class is not a standard class. Its closest standard analogue is perhaps Σ_{i-1}-$TIME(\log n)$.

One can also allow other classes as checking classes. Some classes give analogues of classes above $W[t]$, for example

$$PGC(FPT, ATIME(\log n))$$

or

$$PGC(FPT, SPACE(\log n)).$$

The former class is the same as $AW[*]$ [4]. The latter class contains the former, since $ATIME(O(\log n) \subseteq SPACE(O(\log n))$, but with $SPACE(O(\log n)) \subseteq ATIME(O(\log^2 n))$ as our best result in the reverse direction, one might suspect the two to be unequal. This explains the difficulty of creating "$PSPACE$-analogue" classes as opposed to "AP-analogue" classes [4]. We do not know the relationship of $PGC(FPT, SPACE(O(\log n)))$ to $AW[P]$ or to other studied classes.

References

1. L. Cai and J. Chen, "On the Amount of Nondeterminism and the Power of Verifying," *SIAM J. Computing* 26,3 (1997) 733–750.
2. L. Cai, J. Chen, R.G. Downey, M.R. Fellows, "On the complexity of short computation and factorization," *Archiv. Math. Logic* 36 (1997) 321–337.
3. Y. Chen and J. Flum, "Machine Characterization of the Classes of the W-hierarchy," in *Computer Science Logic: CSL 2003*, Lecture Notes in Computer Science, Springer, 2003, pp. 114–127.
4. Y. Chen, J. Flum and M. Grohe, "Bounded Nondeterminism and Alternation in Parameterized Complexity Theory," in *18th Ann. IEEE Conf. Computational Complexity*, 2003, pp. 18–29.
5. R.G. Downey and M.R. Fellows, *Parameterized Complexity*, Springer, New York, 1999.
6. J. Flum and M. Grohe, "Fixed-Parameter Tractability, Definability, and Model-Checking," *SIAM J. Computing* 31,1 (2001) 113–145.