CONF-960233--2

LA-UR 95-3348

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

TITLE: **COMPLEXITY OF HIERARCHICALLY AND 1-DIMENSIONAL PERIODICALLY SPECIFIED PROBLEMS**

AUTHOR(S): M. V. Marathe, H. B. Hunt III, R.E. Stearns, V. Radhakrishnan

SUBMITTED TO: 13th Symposium on Theoretical Aspects of Computer Science (STACS)
Grenoble, France
Feb 22-24, 1996

DISCLAIMER

MASTER

Los Alamos

Los Alamos National Laboratory
Los Alamos New Mexico 87545

Complexity of Hierarchically and 1-Dimensional Periodically Specified Problems

M.V. Marathe [5] *H.B. Hunt III* [1,2,3] *R.E. Stearns* [1,2,3] *V. Radhakrishnan* [2,4]

August 23, 1995

Abstract

We study the complexity of various combinatorial and satisfiability problems when instances are specified using one of the following specifications: (1) the 1-dimensional finite periodic narrow specifications of Wanke and Ford et al. [FF58, Wa93], (2) the 1-dimensional finite periodic narrow specifications with explicit boundary conditions of Gale [Ga59], (3) the 2-way infinite 1-dimensional narrow periodic specifications of Orlin et al. [Or82a] and (4) the hierarchical specifications of Lengauer et al. [LW87a]. we obtain three general types of results.

First, we prove that there is a polynomial time algorithm that given a 1-FPN- or 1-FPN(BC)-specification of a graph (or a CNF formula) constructs a level-restricted L-specification of an isomorphic graph (or formula). This theorem along with the hardness results proved here provides *alternative and unified* proofs of many hardness results proved in the past either by Lengauer and Wagner [LW92] or by Orlin [Or82a].

Second, we study the complexity of generalized CNF satisfiability problems of Schaefer [Sc78]. Assuming P $\neq$ PSPACE, we characterize *completely* the polynomial time solvability of these problems, when instances are specified as in (1), (2), (3) or (4).

As applications of our first two types of results, we obtain a number of new PSPACE-hardness and polynomial time algorithms for problems specified as in (1), (2), (3) or (4). Many of our results also hold for $O(\log \mathcal{N})$ bandwidth bounded planar instances. The results significantly extend the results in [LW92, Or82a, Sc78].

Keywords: Hierarchical and Periodic Specifications, Computational Complexity, PSPACE-hardness, Satisfiability Problems.

[1]Email addresses of authors: {hunt,res}@cs.albany.edu

[2]Supported by NSF Grants CCR 89-03319, and CCR94-06611.

[3]Department of Computer Science, University at Albany - SUNY, Albany, NY 12222.

[4]Part of the work was done while the author was at SUNY Albany. Current Address: Mailstop 47LA, Hewlett-Packard Company, 19447 Pruneridge Avenue, Cupertino, California 95014-9913. Email: rven@cup.hp.com

[5]Los Alamos National Laboratory P.O. Box 1663, MS M986, Los Alamos NM 87545. Email: madhav@c3.lanl.gov. The work is supported by the Department of Energy under Contract W-7405-ENG-36.

1 Introduction

Large objects with highly repetitive structure can often be represented succinctly. In the past, several methods have been proposed to succinctly represent objects. Here we consider two of these methods, namely (i) the hierarchical specifications [Ga82, LW92, BOW83, RH93] and (ii) the periodic specifications [CM91, IS87, KO91, KS88, Or82a, Wa93].

Hierarchical specifications are used to describe large scale systems with a very regular structure. This is because by using hierarchical specifications, the overall design of an object can be partitioned into the design of a collection of modules; which is a much more manageable task than producing a complete design in one step. Hierarchical specifications have found applications is VLSI layout [HLW92, RH93], finite element analysis, software engineering and datalog queries (see [HLW92, Ma94] and the references therein). Examples of hierarchical specification methodologies previously discussed include [Ga82, LW92, BOW83, RH93]. Periodic specifications can also be used to define large scale systems with highly regular structures. Using periodic specifications, large objects are described as repetitive connection of a basic module. Frequently, the modules are connected in a straight line but the basic modules can also be repeated in two or higher dimensional patterns. Periodic specifications have applications in such diverse areas as transportation planning [Or82a, HLW92, Ma94], parallel programming [HLW92, KMW67] and VLSI design [IS87, HLW92]. They model periodic problems where the constraints or demands for any one period is the same as those for preceding or succeeding periods.

An important feature of both hierarchical and periodic specifications is that they can be much more concise in describing objects than standard specifications. In general, the size of an object can be exponential in the size of its periodic or hierarchical specifications. Since, the complexity of solving a problem is usually measured as a function of the size of the description of problem instance, the complexity of a problem when instances are specified using standard descriptions can be much different than the complexity of the same problem when instances are specified hierarchically or periodically. For example, while 3-COLORING problem [GJ79] is NP-complete when the graphs are represented by an adjacency matrices or by adjacency lists, it is PSPACE-complete when instances are specified hierarchically or periodically [LW92, Or84b]. On the other hand the 2-COLORING problem is solvable in polynomial time *even* when instances are specified using the hierarchical specifications of Lengauer et al. [LW92] or the periodic specifications of Orlin [Or82a].

In this paper, we study the complexity of several combinatorial, graph and generalized satisfiability problems when instances are specified using one of the following specifications: (1) the 1-dimensional finite periodic narrow specifications of Wanke [Wa93], and of Ford and Fulkerson [FF58], (denoted here by 1-FPN-specifications), (2) the 1-dimensional finite periodic narrow specifications of Gale [Ga59] and others (denoted by 1-FPN(BC)-specifications), (3) the 2-way infinite 1-dimensional narrow periodic (sometimes called dynamic) specifications of Karp and Orlin et al. [KMW67, Or82a], (denoted by 1-PN-specifications), or (4) the hierarchical specifications of Lengauer [LW87a], (denoted by L-specifications).

Let Π be a problem posed for instances specified as usually assumed in the literature. For instance, for CNF formulas, the standard specification for instances of satisfiability problems is sets of clauses, with each clause being a set of literals. For problems in graph theory, the adjacency matrix representation or the adjacency list specification of the edges in the graph are standard specifications of the graph. For the rest of the paper we refer to such usual specifications as *standard* specifications. In this paper, we use L-Π, 1-FPN-Π 1-FPN(BC)-Π and 1-PN-Π to denote the problem Π when instances are specified using L-, 1-FPN-, 1-FPN(BC)- 1-PN-specifications respectively. For example, L-3SAT denotes the problem 3SAT when instances are specified using L-specifications and 1-FPN-3SAT denotes the problem 3SAT when instances are specified using 1-FPN-specifications. We also use the term *succinct specifications* to mean one of the above given specifications. *For the rest of the paper we use α to denote*

one of the succinct specifications given above.

2 Summary of results

One important goal of this paper is to provide a better understanding of the efficient inter-translatability of description of objects using various succinct specifications considered. In particular, we show that very simple repetitive structures that can be specified by each of succinct specifications studied in this paper make the problems "hard" to solve. This implies (provides *alternative and unified proofs* for) the PSPACE-hardness results obtained in the past either by Lengauer and Wagner [LW92] or by Orlin [Or82a]. In the past, the hierarchical and periodic specifications have been studied in isolation; our results here point out the close correspondence between the two specifications. The results obtained in this paper extend the results of (i) Schaefer [Sc78], where the complexity of non-succinctly specified satisfiability problems was studied and a complete characterization of the problems was given, (ii) Orlin [Or84b] on the complexity of 1-PN specified problems, (ii) Lengauer and Wagner [LW92] on the complexity of L-specified problems. We discuss these extensions in detail in the subsequent subsections.

2.1 Translation Theorem

We first prove that that given a 1-FPN or 1-FPN(BC) specification of a graph (or a CNF formula) G we can construct a 1-level restricted L-specification of a graph (or a formula) G' which is isomorphic to G. (Theorem 7.1 proved in the Appendix. Henceforth referred to as Translation Theorem.) The translation theorem implies that for all the graph and satisfiability problems Π, the problem Π when specified using a 1-FPN-specification is polynomial time reducible to the problem Π when specified using an L-specification. Thus, given the translation theorem, it suffices to prove lower bound results for 1-FPN-specified problems and upper bound results for L-specified problems. Therefore, for the rest of the paper, whenever possible we prove the hardness results for 1-FPN-specified instances and the easiness results for L-specified instances.

2.2 Complexity of Satisfiability Problems

To understand the results in this subsection, it is useful to first review the relevant definitions from [Sc78]. We assume that the reader is familiar with the problem 3SAT. Let $S = \{R_1, \ldots, R_m\}$ be a finite set of finite arity Boolean relations. An S-*formula* is a conjunction of clauses each of the form $\hat{R}_i(\xi_1, \xi_2, \cdots)$, where $\xi_1, \xi_2, \cdots$ are distinct, unnegated variables whose number matches the rank of $R_i, i \in \{1, \cdots m\}$ and $\hat{R}_i$ is the relation symbol representing the relation R_i. The S-*satisfiability problem* (denoted by SAT(S)) is the problem of deciding whether a given S-formula is satisfiable. The above formulation yields an infinite collection of satisfiability problems which contains the the problems 3SAT and 1-3SAT as special cases. A variation of the SAT(S) problems is the problems SAT_c(S), in which we allow constants 0 or 1 in the input formulas.

Here, we investigate the complexity of the generalized satisfiability problems for succinctly specified instances (denoted as α-SAT(S)) (See Definitions 4.1 and 4.2). Specifically, we **completely characterize** the complexity of the problems α-SAT(S) for every finite set of finite arity boolean relation S. We prove the following theorem.

Theorem 2.1 Let S be a finite set of finite arity boolean relations. Let conditions (a) through (e) be defined as follows.

(a) Every relation in S is 0-valid or every relation in S is 1-valid.

(b) Every relation in S is 1-weakly positive or every relation in S is 1-weakly negative.

(c) Every relation in S is weakly positive or every relation in S is weakly negative.

(d) Every relation in S is affine.

(e) Every relation in S is bijunctive.

1. If S satisfies one of the conditions (b) (d) or (e) then L-SAT$_c$(S) and 1-FPN(BC)-SAT$_c$(S) are in P otherwise L-SAT$_c$(S) and 1-FPN(BC)-SAT$_c$(S) are PSPACE-complete.

2. If S satisfies one of the conditions (a) (b) (d) or (e) then L-SAT(S) and 1-FPN(BC)-SAT(S) are in P. Otherwise, L-SAT(S) and 1-FPN(BC)-SAT(S) are PSPACE-complete.

3. If S satisfies conditions (a), (b), (c), (d) or (e), then the problems 1-FPN-SAT(S) and 1-PN-SAT(S) are in P. Otherwise, the problems 1-FPN-SAT(S) and 1-PN-SAT(S) are PSPACE-complete.

Our work is motivated by the work of Schaefer [Sc78] who gave a complete characterization of the generalized satisfiability problems SAT(S) when instances are specified using standard specifications, and the work of Orlin [Or82a], where he proved the PSPACE-hardness of the problem 1-PN-3SAT. An interesting aspect of our characterization of the problems α-SAT(S) is that for any given finite arity finite set of boolean relations S, the problem α-SAT(S) is either in P or is PSPACE-complete. Thus, our dichotomy result is similar in spirit to the result of Schaefer [Sc78], except that he proved that each of the problems SAT(S) is either in P or is NP-complete. Thus the result significantly extends the work of (i) Schaefer [Sc78], where the complexity of non-succinctly specified satisfiability problems was studied and a complete characterization of the problems was given, and (ii) Orlin [Or82a], where 1-PN-3SAT was proved to be PSPACE-complete. Apart from being of independent interest, the hardness results for the problems α-SAT(S) serve as useful starting points for proving the PSPACE-hardness of various other combinatorial and graph problems specified using one of the specifications α.

Applications

Next, we use the first two types of results together with known "local" reductions between problems to obtain hardness results for various graph theoretic and combinatorial problems, when instances are specified using one of the specification α. Specifically, we show that for most natural problems, if $\Pi_1 \leq \Pi_2$ such that the reduction is local, then the problem α-$\Pi_1 \leq \alpha$-Π_2. Several combinatorial problems Π are known to have local reduction from the problem 3SAT and its variations. These reductions are *lifted* to obtain local reductions from α-3SAT to α-Π. For example we obtain the following result.

Theorem 2.2 The problems α-3SAT, α-HP, α-IS, α-VC, α-3COLORING are PSPACE-hard for $O(\log \mathcal{N})$ bandwidth bounded α-specified planar graphs.

3 Related work and significance

Orlin considered the problem 1-PN-3SAT, and proved that 1-PN-3SAT is PSPACE-complete. Then using known local replacement type reductions from 3SAT to other classical problems Π he showed that the problems 1-PN-Π are PSPACE-complete. The problems shown to be PSPACE-hard include 1-PN-3SAT, 1-PN-Knapsack, 1-PN-HP, 1-PN-3COLORING and 1-PN-3DM. Lengauer and Wagner [LW92] characterize the the complexity of several graph problems when specified hierarchically. They prove the PSPACE-hardness of the problems 3 Coloring (3COLORING), Independent set (IS), Hamiltonian Circuit (HC), Monotone Circuit Value Problem (MCVP), Network Flow (NF) and Alternating Graph Accessibility problem (AGAP).

The proofs of PSPACE-hardness for the problems studied in this paper improve upon known PSPACE-hardness results in several ways. First, for many of problems considered, we give tight lower and upper bound (given the current state of knowledge about computational complexity.) in terms of the space and time required for solving the problems. For example, our results show that the problem 1-PN-3SAT can be decided in non-deterministic linear space. Furthermore, we show that the problem is PSPACE-complete by a $\theta(n)$ size reduction[6] from the membership problem for non-deterministic linearly bounded automaton (LBA). This along with Savitch's result imply that 1-PN-3SAT requires at least $2^{\Omega(n^r)}$ deterministic time to solve. (Assuming that there are languages in NSPACE(n) requiring deterministic time $2^{\Omega(n^r)}$, $0 \leq r \leq 1$.) On the other hand, Orlin's reduction from the membership problem for a non-deterministic LBA to the problem 1-PN-3SAT is a $\theta(n^2)$ size reduction. Such a reduction only implies that 1-PN-3SAT takes deterministic time $2^{\Omega(n^{r/2})}$ to solve the problem. Similar lower bound results hold for the space required to solve L-specified problems. For example, Lengauer and Wagner have a $\theta(n^4)$ sized reduction from the languages in DSPACE(n) to the L-MCVP (monotone circuit value problem). Here we show that L-MCVP is PSPACE-complete by a $\theta(n^2)$ sized reduction from from the languages in DSPACE(n). Thus while our reduction implies a lower bound of $2^{\Omega(n^{r/2})}$ (Assuming that there are languages in DSPACE(n) requiring deterministic time $2^{\Omega(n^r)}$, $0 \leq r \leq 1$.) for solving L-MCVP, the reduction of Lengauer and Wagner only implies a lower bound of $2^{\Omega(n^{r/4})}$ for solving L-MCVP. Similarly the PSPACE-hardness of L-HC is by a $\theta(n^4)$ sized reduction from the languages in DSPACE(n). This reduction only implies a lower bound of $2^{\Omega(n^{r/4})}$ for solving L-MCVP. In contrast, we can show that L-HC requires $2^{\Omega(n^{r/2})}$ deterministic time to solve the problem. In several cases, we can also give algorithms that solve the problem in time that matches the lower bounds proved here.

Second, the PSPACE-hardness results for most of the problems considered here hold for $O(\log \mathcal{N})$ bandwidth bounded planar instances. No hardness results of any kind were presented for problems restricted to planar instances specified using any of the specifications α. Also, previously, the only known PSPACE-hardness results for bandwidth bounded succinctly specified instances were the PSPACE-hardness of L-3COLORING and L-IS [LW92]. Furthermore, in [LW92] the proof of PSPACE-hardness of L-HC and L-MCVP is from QBF. It is not clear how to extend these reductions so as to extend the PSPACE-hardness results for $O(\log \mathcal{N})$ bandwidth bounded planar instances. In contrast, oour PSPACE-hardness results can be very easily extended to obtain results for $O(\log \mathcal{N})$ bandwidth bounded planar instances.

Third, other than in the work of Orlin [Or82a], the complexity of satisfiability problems for succinct specifications has not been not considered in the past. In particular, except for references in our papers [MH+93a, MH+94] α-specified CNF formulas and their satisfiability problems have **not** appeared previously in the literature.

The rest of the paper consists of selected proof sketches. Additional proofs appear in the Appendix. For more details, we refer the reader to the complete version of the paper.

4 Preliminaries

Due to the lack of space, we refer the reader to the work of [LW92, HLW92, Le89, MH+94, Or82a, Or84b] for definition of L-, 1-PN-, 1-FPN- and 1-FPN-specified graphs.

[6] We say that $A \subseteq \Gamma^*$ is PSPACE-complete **via a size** $L(n)$ reduction if A is PSPACE-complete and for any $B \subseteq \Delta^*$ such that B is decidable (non)-deterministically in space $S(n) = n$, $B \leq_p A$ via some function f such that for all $w \in \Delta^*$, $|f(w)| \leq cL(|w|)$, where c is a constant depending on B. If $f(n) = O(n)$, we say that the reduction is a linear size reduction.

4.1 Hierarchically specified formulas

Here, we introduce the concept of hierarchically specified S-formulas and the associated generalized satisfiability problems. Such formulas are built by defining larger S-formulas in terms of smaller S-formulas. Just as L-specifications can represent objects that are exponentially larger than the specification, hierarchically specified S-formulas can be exponentially larger than the size of their specification. This concept has been discussed briefly in [MH+93a, MH+94].

Definition 4.1 An instance $F = (F_1(X^1), \ldots, F_{n-1}(X^{n-1}), F_n(X^n))$ of L-SAT(S) is of the form

$$F_i(X^i) = (\bigwedge_{1 \le j \le l_i} F_{i_j}(X^i_j, Z^i_j)) \bigwedge f_i(X^i, Z^i)$$

for $1 \le i \le n$ where f_i are S-formulae, $X^n = \phi$, X^i, X^i_j, Z^i, Z^i_j, $1 \le i \le n-1$, are vectors of boolean variables such that $X^i_j \subseteq X^i$, $Z^i_j \subseteq Z^i$, $0 \le i_j < i$. Thus, F_1 is just a CNF formula. An instance of L-SAT(S) specifies a 3CNF formula $E(F)$, that is obtained by expanding the F_j, $2 \le j \le n$ as macros where the variables Z's introduced in any expansion are considered distinct. The problem L-SAT(S) is to decide whether the formula $E(F)$ specified by F is satisfiable.

The set of variables Z^i in the corresponding F_i are called *explicit variables*. Let n_i be the total number of variables used in F_i (i.e. $|X^i|+|Z^i|$) and let m_i be the total number of clauses in F_i. The size of F, denoted by $size(F)$, is equal to $\sum_{1 \le i \le n}(m_i n_i) = O(MN)$ where $M = \sum_i m_i$ and $N = \sum_i n_i$. Given a formula $E(F)$ specified by a hierarchical specification F, $BG(E(F))$ denotes the bipartite graph associated with $E(F)$. We use $G_{BG(E(F))}$ to denote the hierarchical specification of $BG(E(F))$. In [MH+93a] we outline a transformation which on input F constructs $G_{BG(E(F))}$ in polynomial time. We denote the size of $E(F)$ (using standard specifications) by $\mathcal{N}$.

The above definition yields as special cases analogues of many of the well known satisfiability problems such as 1-3SAT, NAE3SAT, etc. denoted by L-1-3SAT and L-NAE3SAT respectively. For instance, the problem L-1-3SAT is defined analogously as in Definition 4.1 except that each f_i is a 3CNF formula, which is true precisely when one of the literals is assigned a value 0.

Example: Let $F = (F_1(x_1, x_2), F_2(x_3, x_4), F_3)$ be an instance of L-3SAT where each F_i is defined as follows:

$$F_1(x_1, x_2) = (x_1 + x_2 + z_1) \wedge (z_2 + z_3)$$

$$F_2(x_3, x_4) = F_1(x_3, z_4) \wedge F_1(z_4, z_5) \wedge (z_4 + z_5 + x_4); \quad F_3 = F_2(z_8, z_7) \wedge F_1(z_7, z_6)$$

The formula $E(F)$ denoted by F is given by $(z_7 + z_6 + z^1_1) \wedge (z^1_2 + z^1_3) \wedge (z_8 + z_4 + z^2_1) \wedge (z^2_2 + z^2_3) \wedge (z_4 + z_5 + z^3_1) \wedge (z^3_2 + z^3_3) \wedge (z_4 + z_5 + z_7)$.

4.2 Periodically specified formulas

Let $U = \{u_1, \ldots, u_n\}$ be a finite set of variables (referred to as static variables). $U^\infty = \{u_k(i) : 1 \le k \le n,\ i \in \mathbf{Z}\}$. $U^M = \{u_k(i) : 1 \le k \le n,\ i \in \{0, 1, 2, \cdots, M\}\}$. (In our proofs, variable $u_k(i)$ denotes the variable u_k at grid point i.) A literal of U is an element of $\{u_1, \ldots, u_n, \overline{u_1}, \ldots, \overline{u_n}\}$. If w is a literal of U, then $w(i)$, is a literal of U^∞. Let $C(i)$ be a parameterized conjunction of 3 literal clauses such that each clause in $C(i)$ consists of variables $u_k(i), u_k(i+1)$ with the constraint that at least one variable is of the form $u_k(i)$. We refer to the clauses $C(i)$ as static narrow clauses. ($C(i)$ is called narrow because for all $(w_1(i_1) \vee w_2(i_2) \vee w_3(i_3)) \in C(i)$, $|i_s - i_r| \le 1$ for $1 \le r \le s \le 3$.) Let $\Gamma = (U, C(i), M)$. Let $C^\infty = \bigwedge_{i=0}^{i=\infty} C(i)$. Then C^∞ is the 3CNF formula specified by Γ. Given U^M and C^∞, let C^M be a subset of C with the following property: for each clause $(w_1(i_1) \vee w_2(i_2) \vee w_3(i_3)) \in C^M$, $w_1(i_1), w_2(i_2), w_3(i_3)) \in U^M$.

It is useful to imagine a narrow periodically specified formula Γ^∞ as being obtained by placing a copy of the variable set U at each integral point (also referred to as lattice point) on the X-axis (or the time line). Furthermore, assume that the clauses $C(t)$ are placed at time t. With this notation, we can now refer to variables at time t as the set of variables $U(t)$ and clauses at time t to clauses at time t

Definition 4.2 A 1-dimensional infinite (finite) periodic narrow specification (denoted by 1-PN (FPN)-specification) of a 3CNF formula $F^\infty(U^\infty, C^\infty)$ ($F^M(U^M, C^M)$) is given by ($\Gamma = (U, C(i))$, ($\Gamma = (U, C(i), M)$), where, U is a a finite set of variables, $C(i)$ is a collection of static narrow 3 literal clauses. (In case of finite specifications M is non-negative integers specified in binary.) The size of the specification denoted by $size(\Gamma) = |U| + |C(i)|$. (In case of finite specifications $size(\Gamma) = |U| + |C(i)| + bits(M)$, where $bits(M)$ denote the number of bits used to represent M.) We denote the size of F^M specified using standard specifications) by $\mathcal{N}$.

The problem 1-PN-3SAT (problem 3SAT specified using 1-FPN-specifications) is the problem of determining if a 3CNF formula $F^\infty(U^\infty, C^\infty)$ specified by $\Gamma = (U, C(i))$ is satisfiable.

Similarly, the problem 1-FPN-3SAT (problem 3SAT specified using 1-FPN-specifications) is the problem of determining if a 3CNF formula $F^M(U^M, C^M)$ specified by $\Gamma = (U, C(i), M)$ is satisfiable.

For each finite set S of finite arity Boolean relations, it is straightforward to extend the above definition so as to define the problems 1-PN-SAT(S) and 1-FPN-SAT(S) and hence we omit these definitions. Observe that 1-FPN-specified graphs or formulas can be exponentially larger than their input specifications.

Example 4: Let $F = (U, C(i), 3)$ be an instance of 1-FPN-3SAT where the set of static clauses are given by $(x_1(0)+x_2(0)+x_3(0)) \wedge (x_1(1)+x_3(0)) \wedge (x_3(1)+x_2(0))$. The set of variables $U = \{x_1, x_2, x_3\}$. The formula $F^3(U^3, C^3)$ denoted by Γ is given by

$$(x_1(0) + x_2(0) + x_3(0)) \wedge (x_1(1) + x_3(0)) \wedge (x_3(1) + x_2(0)) \bigwedge$$

$$(x_1(1) + x_2(1) + x_3(1)) \wedge (x_1(2) + x_3(1)) \wedge (x_3(2) + x_2(1)) \bigwedge$$

$$(x_1(2) + x_2(2) + x_3(2)) \wedge (x_1(3) + x_3(2)) \wedge (x_3(3) + x_2(2)) \bigwedge \ (x_1(3) + x_2(3) + x_3(3)) \ \blacksquare$$

Some instances of problems arising in practice have a periodic specification of the graph or a formula along with explicit initial and final conditions. We call such periodic specifications as periodic specifications with boundary conditions (BC). Observe that for 1-PN-specifications boundary conditions do not make any sense since the expanded graph or the formula is infinite in both directions. Hence boundary conditions are used to augment only 1-FPN-specifications.

5 PSPACE-completeness of α-3SAT

Theorem 5.1 (1) 1-FPN-3SAT is in NSPACE(n).
(2) There is a linear size quasi linear time reduction from the membership problem for non-deterministic LBA to the problem 1-FPN-3SAT. Thus 1-FPN-3SAT is PSPACE-complete.

Proof:
Part (1): We first show that the problem is in NSPACE(n) (and therefore in DSPACE(n^2)). To show this, observe that given a 1-FPN-specification $\Gamma = (G(V, E), m)$, a Turing machine needs to maintain at any given time t only assignments to variables at time t and time $t + 1$. This is because without loss of generality we can assume that a clause specified at time t contains variables defined at time t and $t + 1$. Hence a polynomial non deterministic linearly space bounded Turing machine can verify that the instance of 1-FPN-3SAT is satisfiable as follows. At each step t it guesses an assignment to the

variables at grid point $t+1$. It also remembers the assignment to the variables at grid point t. Using these values it verifies that the clauses at time t are indeed satisfied. This proves that given an instance Γ of 1-FPN-3SAT, we can decide in non-deterministic *linear* space if the formula G^m is satisfiable.
Part (2): Next, we prove 1-FPN-3SAT is PSPACE-hard. Given a non-deterministic LBA, M, with input x (where $|x| = n$), we construct an instance of 1-FPN-3SAT $F(x)$ such that $size(F(x) = O(n)$, and x is accepted by M if and only if $F(x, t, t+1)$ is satisfiable. The reduction consists of two phases.
Phase 1: In the first phase, we start with the given LBA M with input $x = (x_1, \ldots, x_n)$ and construct a new LBA M_1 which simulates M on x with the following additional properties that

1. if the LBA M does not accept x then each computation of M_1 on x halts within 2^{cn} moves, else
2. if the LBA M accepts x then M_1 has a cycling computation, where the length of an ID never exceeds $O(|x|)$.

M_1 can be constructed easily by adding an auxiliary clock to serve as a counter. M_1 now just simulates M. If M enters a final configuration, then M_1 repeats this configuration. Its clear that M_1 accepts x if and only if M accepts x and their number of accepting computations is the same. Moreover it is easy to see that M_1 has the two desired properties above.
Phase 2: The second phase consists of constructing an instance $\wedge_{t=0}^{N} F(x, t, t+1)$ of 1-FPN-3SAT by a polynomial time reduction from M_1. Now we know that each ID of the Turing machine M_1 is of length $O(n)$, where n is the size of the input. Since M_1 is non-deterministic LBA we need to consider only 2^{Dn} different ID's for our reduction. (Here D is an appropriately chosen constant.) We can choose an encoding of states and symbols of M_1 into words in $\{0,1\}^*$ so that every ID of M_1 will consist of $c_1 \cdot n$ boolean variables where $n = |x|$ and c_1 is a constant independent of x. Let $ID(t)$ denote the ID of the Turing machine at time t. We also have a set of $Dn+1$ boolean variables encoding a counter $\gamma(t)$. The counter values range from 0 to 2^{Dn}. The formula $F(x,t,t+1) = f_1(x,t,t+1) \wedge f_2(x,t,t+1) \wedge f_3(x,t,t+1)$. We discuss each of the $f_i(t, t+1)$, $1 \leq i \leq 3$.

1. The formula $f_1(x, t, t+1)$ encodes a counter which is given by

$$\gamma(t+1) = (\gamma(t) + 1) \pmod{2^{Dn} + 1}.$$

   The intended meaning of the equation is that the counter resets after every $2^{Dn} + 1$ time units. It is easy to see that the counter can be simulated by a CNF formula, in which each clause has variables that are no more than one time unit apart.

2. The formula $f_2(x, t, t+1)$ enforces the condition that when the counter value is 0, the variables $X(t)$ encode the starting ID of the Turing machine. Here, $start(ID(t))$ denotes a CNF formula which checks if $ID(t)$ is the initial ID of the machine. Thus $f_2(t, t+1)$ is a 3CNF formula which encodes the implication $(\gamma(t) = 0) \Rightarrow start(ID(t))$.
   Again, we can verify that $f_2(x, t, t+1)$ can be written as a 3CNF formula in polynomial time. Again by standard techniques it follows that the formula $f_2(x, t, t+1)$ is of size $O(n)$.

3. The formula $f_3(x, t, t+1)$ is needed to ensure that starting at the second ID, each subsequent ID of M_1 follows from the previous ID by using the transition function of M_1. (Recall that the notation $(X \vdash_M^j Y)$ means that machine M, starting with ID X, can produce the ID Y in exactly j steps.) Thus $f_3(t)$ is a 3CNF formula encoding the following implication.

$$(1 \leq \gamma(t) \leq 2^{Dn}) \Rightarrow (ID(t-1) \vdash_M ID(t))$$

   The function $(ID(t) \vdash_M ID(t+1))$ can be expressed by 3CNF formula whose sizes are linear in n as shown in [Hu73a]. Moreover the 3CNF formula depends on the current value of the counter.

Hence the CNF formula $f_3(x,t,t+1)$ is narrow periodic formula, since the clauses at time t would contain variables only from times t and $t+1$.

The expanded finite periodic 3SAT instance is $\bigwedge_{t=0}^{N} F(x,t,t+1)$, where $N = 2^{2Dn}$. We now verify that $\bigwedge_{t=0}^{N} F(x,t,t+1)$ is an instance of 1-FPN-3SAT.

We now prove the correctness of our reduction. If the Turing machine M accepts x then we know that M_1 has a cycling computation. Hence by setting $d_t = 0$, we can ensure that $f_2(0)$ is satisfied. The consistency condition now forces the formula to be satisfiable

Conversely, assume that the formula is satisfiable. Since D (and hence N) are suitably large integers, it is guaranteed that the simulation must be carried out for enough steps so that the Turing machine M_1 goes through the sequence $d_t = 0, d_t = 1, d_t = 2, \cdots d_t = 2^{Dn}$. This implies that the formulas $f_2(t)$ and $f_3(t)$ would be true from then on and therefore the Turing machine M accepts x. ■

As our next corollary shows, 1-FPN-3SAT is even when restricted to formulas with bandwidth $O(\log \mathcal{N})$. Recall that $\mathcal{N}$ denotes the size of the encoding of the expanded formula using standard encodings.

Corollary 5.2 1-FPN-3SAT is PSPACE-complete even when restricted to formulas with bandwidth $O(\log \mathcal{N})$.

Proof: The proof follows by observing that the following numbering scheme yields a $O(\log \mathcal{N})$ bandwidth layout. Let $C(i)$ have p clauses. Then number the clauses at time t using numbers from pt to $(p+1)t$. The numbering of clauses at a given time t can be carried out in any arbitrary order. ■

Remark: The above corollary in conjunction with local replacement type reductions between problems specified using standard specifications can be used to prove that several classical graph problems are PSPACE-hard even for $O(\log \mathcal{N})$ bandwidth bounded graphs specified using *either* 1-FPN-specifications or strongly 1-level-restricted L-specifications (since the L-specification obtained by the translation theorem represents an isomorphic graph (formula)).

Next, we discuss the PSPACE-hardness of the problem 1-FPN(BC)-3SATWP. 1-FPN-3SATWP (problem 3SATWP specified using 1-FPN-specifications) is the problem of determining if a 3CNF formula $F^M(U^M, C^M)$ which contains at most one negated literal per clause and is specified by $\Gamma = (U, C(i), M)$ is satisfiable.

Theorem 5.3 The problems 1-FPN(BC)-3SATWN and 1-FPN(BC)-3SATWP are PSPACE-complete by a linear size quasi-linear time reduction from the membership problem for deterministic LBA.

Proof: We prove the result for 1-FPN(BC)-3SATWN. The proof for 1-FPN(BC)-3SATWP is similar. The membership of the problem 1-FPN(BC)-3SATWN in NSPACE(n) follows since the problem 1-FPN(BC)-3SAT is in NSPACE(n). To prove that 1-FPN(BC)-3SATWN is PSPACE-hard we give a linear size reduction from the acceptance problem of a bf deterministic linearly bounded machine. Let $ID(t)$ denote the instantaneous description of a TM at time time t. Given a deterministic LBA M and input x such that $|x| = n$ we create an instance $(F(x,t,t+1), m)$ of 1-FPN(BC)-3SATWN, such that $\wedge_{t=0}^{t=N} F(x,t,t+1)$ is satisfiable if and only if M accepts x. The reduction is similar to one presented for 1-FPN-3SAT. The formulas encoding $(ID(t) \vdash ID(t+1))$, and $start(ID(t))$ can be represented by a weakly negative formula as in [JL77] for UNIT. By negating all literals, we can obtain a weakly positive formula proving the PSPACE-hardness of L-3SATWP. Let $a \in \{\#\} \cup T \cup (S \times T)$, where T denotes the tape symbols and S denotes the set of states. Let $P_i^a(t)$ be a boolean variable which means that the contents of i^{th} tape cell at time t is a. Observe that for a given value of t the number of variables $P_i^a(t)$ is $O(n)$. The formula $g(x,0)$ is now represented as

$$g(x,0) \equiv \left(P_1^{(q_0,a_1)}(0) \wedge P_2^{a_2}(0), \ldots \wedge P_n^{a_n}(0)\right) \bigwedge_i \left(\bigwedge_{a \neq b} (\overline{P_i^a(0)} \vee \overline{P_i^b(0)}) \right)$$

The first part represents the condition that the first ID corresponds to the input and the second part represents the condition that one position cannot contain two distinct symbols. Hence, the above formula represents the condition that the first ID is correct. Also observe that since the number of tape symbols are constant, the size of $g(x,0)$ is linear in the size of the input.

Next, we represent $(ID(t) \vdash_M ID(t+1))$ as follows: Let $f : (T \cup (S \times T))^3 \rightarrow T \cup (S \times T)$ be the finite function such that if positions $i-1$, i and $i+1$ of the $ID(t)$ contain a, b and c respectively, then position i of the $ID(t+1)$ must contain $f(a,b,c)$. The determinism of M ensures that f is single valued. We express the requirement that $ID(t+1)$ is appropriately determined by $ID(t)$ as follows:

$$g(x,t,t+1) \equiv \bigwedge_{i=1}^{i=n} \bigwedge_{a,b,c \in T} \Big(\big(P^a_{i-1}(t) \wedge P^a_i(t) \wedge P^a_{i+1}(t) \big) \Rightarrow P_i^{f(a,b,c)}(t+1) \Big)$$

which can be written as an equivalent weakly negative formula as follows:

$$g(x,t,t+1) \equiv \bigwedge_{i=1}^{i=n} \bigwedge_{a,b,c \in T} \Big(\overline{P^a_{i-1}(t)} \vee \overline{P^a_i(t)} \vee \overline{P^a_{i+1}(t)} \vee P_i^{f(a,b,c)}(t+1) \Big)$$

Now, each four literal clauses replaced by equivalent three literals clauses by adding new auxiliary variables $T_i^{f(a,b,c)}(t)$. Therefore $g(x,t,t+1)$ can now be rewritten as

$$\bigwedge_{i=1}^{i=n} \bigwedge_{a,b,c \in T} \Big(\big(\overline{P^a_{i-1}(t)} \vee \overline{P^a_i(t)} \vee T_i^{f(a,b,c)}(t) \big) \wedge \big(\overline{T_i^{f(a,b,c)}(t)} \vee \overline{P^a_{i+1}(t)} \vee P_i^{f(a,b,c)}(t+1) \big) \Big)$$

$F(x,t,t+1) = g(x,0) \cup g(x,t,t+1)$. Now let $N = 2^{2Dn}$. The instance obtain as a result of the reduction is $(F(x,t,t+1), N)$ The expanded formula is now given as

$$F^N(x) = g(x,0) \wedge \bigwedge_{t=0}^{N} g(x,t,t+1).$$

Again observe that since the number of tape symbols are constant and i varies from 1 to n, the size of the formula $g(x,t,t+1)$ is linear in the size of the input. Also, observe that representation of N is of size $O(n)$. Therefore the size of the instance obtained as a result of the reduction is $O(n)$. Now N is suitably chosen large integer so that the simulation can be carried out for enough number of states. This completes the proof of the theorem. ■

We note by a direct extension of this proof, we obtain the PSPACE-hardness of the problem 1-FPN(BC)-MCVP. Also observe that the monotone circuit value problem makes sense only for 1-FPN(BC)-specifications. Alos observe that the hardness results discussed above hold for $O(\log \mathcal{N})$ bandwidth bounded instances.

5.1 Hardness of α-SAT(S)

We now show that when Rep(S) denotes the set of all boolean relations, α-SAT(S) is PSPACE-complete. Recall that $Rep(S)$ is the set of relations that are representable by existentially quantified S-formulas with constants. $Rep_{NC}(S)$ is the set of relations that are representable by existentially quantified S-formulas without constants.

Theorem 5.4 If Rep(S) is the set of all finite arity boolean relations, then
(a) the problem 1-FPN-SAT$_c$(S) is PSPACE-complete. (b) The problems 1-FPN-SAT(S) and 1-PN-SAT(S) are in P if each relation in S is 0-valid or each relation in S is 1-valid, otherwise the problems 1-FPN-SAT(S) 1-PN-SAT(S) are PSPACE-complete.

Proof: First, we give a polynomial time reduction from 1-FPN-3SAT to 1-FPN-SAT$_c$(S) as follows: Since Rep(S) is the set of all finite arity boolean relations, for each such S and 3CNF clause c, there is an existentially quantified (but not necessarily constant-free) S-formula f_c such that $c = [f_c]$. Again up to an easy renaming of variables, there are only fourteen such formulae.

Let F be an instance of 1-FPN-3SAT. Obtain a static formula by replacing every clause c with the existentially quantified (but not necessarily constant-free) S-formula f_c such that $c = [f_c]$. For each of these clauses, let f'_c be the S-formula (again, possibly with of constants) resulting from f_c by deleting all quantifiers after making sure that all quantified variables are distinct from each other and from all free variables. Without loss of generality, we assume that, for all clauses c, all variables of f'_c, that are not variables of f, are local to f'_c. We have now obtained an instance F' of 1-FPN-SAT$_c$(S), which is satisfiable if and only if F was satisfiable.

Next, we now reduce 1-FPN-SAT$_c$(S) to 1-FPN-SAT(S). Let S be a set of nonempty logical relations. Then as shown in [Sc78] one of the following holds: (1) Every relation in S is 0-valid. (2)Every relation in S is 1-valid. (3) $[x]$ and $[\overline{x}]$ are contained in Rep_{NC}. (4) $[x \not\equiv y] \in \text{Rep}_{NC}(S)$. Hence the proof consists of a case analysis depending on which of above conditions is satisfied by S.

Cases (1) and (2): A logical relation R is **0-valid** if $(0, \ldots, 0) \in R$. A logical relation R is **1-valid** if $(1, \ldots, 1) \in R$. Therefore in both these cases it is easy to see that 1-FPN-SAT(S) and 1-PN-SAT(S) are polynomial time solvable.

Case (3): $[x]$ and $[\overline{x}]$ are contained in Rep_{NC}.

In this case, we modify F' to obtain F'' as follows. We introduce auxiliary variables and replace all occurrences of 0 by $[x(t)]$ and replace all occurrences of 1 by $[\overline{x(t)}]$ and replace all existentially quantified variables by new variables.

Case (4): $[x \not\equiv y] \in Rep_{NC}(S)$.

In this case, each relation in S is complementive. We modify F' as follows to obtain F''. We introduce two new variables $x(t)$ and $y(t)$ and replace all occurrences of 0 by $x(t)$ and replace all occurrences of 1 by $y(t)$. We then add the S-formulas $[x_t \not\equiv y_t] \wedge [x_t \not\equiv r_t][r_t \not\equiv x_{t+1}] \wedge [y_t \not\equiv s_t][s_t \not\equiv y_{t+1}]$. to the static S-formula, and replacing all existentially quantified variables by new variables $z_1(t), \ldots z_p(t)$. Since F' is complementive, F'' is also complementive.

Therefore, we have now obtained an instance F'' of 1-FPN-SAT(S), which is satisfiable if and only if F' is satisfiable. Clearly the hardness of 1-FPN-SAT(S) implies the hardness of the corresponding problems 1-PN-SAT(S). ■

6 Polynomial time solvable subcases

6.1 Solving 1-FPN-3SATWN and 1-FPN-3SATWP

We now consider the problems 1-FPN3SATWP and 1-FPN3SATWN. The polynomial time solvability of these problems demonstrates the relative hardness of problems (assuming P $\neq$ PSPACE) when specified using 1-FPN or 1-FPN(BC) specifications. In particular, while the problems 1-FPN(BC)-3SATWP and 1-FPN(BC)-3SATWN are PSPACE-hard the problems 1-FPN-3SATWN and 1-FPN-3SATWP are in P. (In the appendix we show how to solve L-specified satisfiability problems when each relation is bijunctive, 1-weakly positive or 1-weakly negative.) The algorithm for solving the problem 1-PN-3SATWP is relatively easy, due to the following observation: If an instance of 1-PN-3SATWP is satisfiable ther exists a satisfying assignment that assigns the same value to all the copies of a given variable in the static formula. This is intuitively true because, in case of 1-PN-3SATWP, whenever we have a clause of the form $x_i(t)$ or $x_i(t+1)$, we have that all the copies of the variable x_i have to be set to true (A similar argument applies for negated clauses). Any algorithm for solving this problem must be more subtler. This is because in the case of finite instances while a clause of the form $x_i(t)$ implies that x_i

Heuristic ALG-AFFINESAT
Input: *An L-specification* $F = (F_1, \ldots, F_n)$ *of a SAT(S) formula* $E(F)$, such that every relation in S is affine.
Output: **Yes** *if and only if* $E(F)$ *is satisfiable.*

1. Repeat the following steps for $1 \leq i \leq n$.

   (a) Let $F_{i_1}, \ldots, F_{i_{r_i}}$ denote the non-terminals called in F_i. Replace each of the F_{i_j} by the smaller set of equations $F^b_{i_j}$ that has been already computed.
   Remark: The sizes of F_{i_j}, $1 \leq j \leq r_i$ is $O(n^2_{i_j})$.

   (b) Let $G_i(X^i)$ represent the set of equations over $X^i \cup Z^i$ obtained as a result of substitution. Here X^i is the set of pin variables of G_i and Z^i is the set of explicit variables of G_i.

   (c) Using Gaussian elimination, eliminate all the variables which are not in X^i, to obtain a set of equations F^b_i only using variables in X^i.
   Remark: The number of independent equations obtained is no more than $|X^i|$ and each equation can have at most all the variables in X^i. Hence the size of F^b_i is $O(n^2_i)$.

   (d) Check if F^b_i is consistent. If not then output **unsatisfiable** and Stop.

2. F is satisfiable if F^b_n is consistent.

Algorithm 4: Details of the algorithm to solve L-SAT(S) when every relation in S is affine.

is set to true for time periods, it is not necessarily true if there is a clause of the form $x_i(t+1)$. Such a clause does imply anything about the value of the variable $x(0)$ in a satisfying assignment of the formula. The following example shows that the only way the expanded formula is satisfied is to assign different values to the copy of a particular variable.

Example 5: Let $F = (U, C(t, t+1), 2)$ be an instance of 1-FPN-3SAT where the set of static clauses are given by $(x_1(t) + x_2(t+1)) \wedge (\overline{x_2(t)}) \wedge (x_2(t) + x_1(t+1))$. The set of variables are $U = \{x_1, x_2, x_3\}$. The formula $F^1(U^2, C^2)$ denoted by Γ is given by

$$(\overline{x_1(0)} + x_2(1)) \wedge (\overline{x_2(0)}) \wedge (x_1(1) + x_2(0)) \bigwedge$$

$$(\overline{x_1(1)} + x_2(2)) \wedge (\overline{x_2(1)}) \wedge (x_1(2) + x_2(1)) \bigwedge (\overline{x_2(2)})$$

By inspection it is clear that whenever $x_1(0) = x_1(1) = x_1(2)$ and $x_2(0) = x_2(1) = x_2(2)$ cannot satisfy the formula F^2. But, $x_2(0) = x_2(1) = x_2(2) = False$, $x_1(0) = True$ and $x_1(1) = x_1(2) = False$ satisfies the formula F^2. ■

Example 5 suggests that for designing a polynomial time algorithm for solving 1-FPN-3SATWP should distinguish the between a copy of the variable at time $t = 0$ and the copy of the same variable at time $t \geq 0$. Algorithm 2 outlines the method to solve the problem. In describing the algorithm and its proof, we use $v[x_i(t)]$ to denote the value assigned to the variable $x_i(t)$. The proof of correctness is omitted due to lack of space.

6.2 Solvability of α-SAT(S) for affine relations

We consider the complexity of α-SAT(S) when every relation in S are affine. The logical relation R is **affine** if $R(x_1, x_2, \ldots)$ is logically equivalent to some system of linear equations over the two-element field $\mathbf{Z}_2$. Algorithm 4 gives the method for solving this problem in polynomial time.

Acknowledgement: We thank Professors Ken Regan and Eric Allender for constructive suggestions on an earlier draft of this paper. We also thank Sandeep Shukla for his help during the course of writing this paper.

Heuristic ALG-1-FPN-3SATWP
Input: *A 1-FPN-specification* $(F(U, C(t,t+1), m)$ *of a 3SATWP formula* F^m. Here $U = \{x_1, \ldots, x_n\}$ denotes the set of variables in the static formula.
Output: **Yes** *if and only if* F^m *is satisfiable.*

1. *flag = 0* and *satisfy = -1*. Let $\mathcal{D}(t) = \phi$. Also let $r = \{t, t+1\}$ and $j = 0$.
2. **Repeat until** *flag =1*

   (a) If $x_i(r) \in C(t,t+1)$ and $\overline{x_i(r)} \in C(t,t+1)$, then set *flag = 1* and *satisfy = 0*. Return to Step 4.

   (b) If $x_i(t+1) \in C(t,t+1)$ and $\overline{x_i(t+1)} \in C(t,t+1)$, then set *flag = 1* and *satisfy = 0*. Return to Step 4.

   (c) **If** all clauses in F contain a positive literal **or** all clauses in F contain a negative literal **then** *satisfy = 1* and *flag = 1*. Go to Step 3.

   Remark: An assignment of the form $v[x_i(r)] = 1$, $\forall x_i \in C(t,t+1)$, $t \leq r \leq t+1$ satisfies the clauses in $\wedge_{t=1}^{t=m} C(t,t+1)$. Therefore, we only need to verify that $\mathcal{D}(t)$ is satisfiable.

   (d) **else**

      i. **If** there is a single literal clause $\overline{x_i(t)}$, $1 \leq i \leq n$ then **do**

         A. $t \leq r \leq t+1$, set $v[x_i(r)] = 0$.

         B. If there is a clause in $\mathcal{D}(t)$ that contains $\overline{x_i(r)}$, then delete the clause. Similarly, delete all occurrences of $x_i(r)$ from all the clauses in $\mathcal{D}(t)$.

      ii. **else**

         A. Pick a single literal clause $\overline{x_i(t+1)}$.
         Remark: Existence of such a clause follows from (i) the definition of 3CNF weakly positive formula, and (ii) Conditions for executing Steps 2(c) and 2(d) are not satisfied.

         B. Set $v[x_i(t+1)] = 0$.
         Remark: Existence of a clause $\overline{x_i(t+1)}$ implies that for the formula F^∞ to be true, $0 \leq t \leq m$, $v[x_i(t+1)] = 0$. This does not force an assignment for $x_i(0)$.

         C. Set $\mathcal{D}(t) = \mathcal{D}(t) \cup Cl_{x(t)}$, where $Cl_{x(t)}$ denotes the set of clauses containing $x(t)$. If there is a clause in $\mathcal{D}(t)$ that contains $\overline{x_i(t+1)}$, then delete the clause. Similarly, delete all occurrences of $x_i(t+1)$ from all the clauses in $\mathcal{D}(t)$.
         Remark: $0 \leq t \leq m$, $\overline{x_i(t+1)} \in C^\infty$ implies that if F^∞ is satisfiable, then $1 \leq t \leq m$, $v[x_i(t)] = 0$.

      iii. Let $N_{x_i}(t,t+1) \subseteq C(t,t+1)$ denote the set of clauses containing $\overline{x_i(r)}$. Also, let $P_{x_i}(t,t+1) \subseteq C(t,t+1)$ denote the set of clauses containing $x_i(r)$. Modify $P_{x_i}(t,t+1)$ by deleting the literal $x_i(r)$ from each of the clauses in $P_{x_i}(t,t+1)$. Let $P'_{x_i}(t,t+1)$ denote the resulting set of clauses.

      iv. $C(t,t+1) = (C(t,t+1) - N_{x_i}(t,t+1) - P_{x_i}(t,t+1)) \cup P'_{x_i}(t,t+1)$.
      Remark: Delete all clauses containing the occurrence of the variable $\overline{x_i(r)}$ and for all clauses which contain $x_i(r)$ delete it from the clause.

      v. $j = j+1$
      Remark: The formula F^∞ as a result of modification is given by

$$F_j^\infty = F_{j-1}^\infty (v[x_i(t)] = 1,\ 1 \leq t \leq m) \cup \mathcal{D}(t) - (\wedge_t N_{x_i}(t,t+1)) - (\wedge_t P_{x_i}(t,t+1)) \cup \left(\wedge_t N'_{x_i}(t,t+1)\right)$$

   (e) If the formula $F = \phi$, then set *satisfy = 1* and *flag = 1*.

3. Instantiate $t = 0$ for all the variables in the clauses in $\mathcal{D}(t)$. Check if the clauses in $\mathcal{D}(t)$ are satisfiable. If $\mathcal{D}(0)$ is satisfiable then *satisfy = 1* else *satisfy = 0*.
   Remark: All the clauses in $\mathcal{D}(0)$ contain variables only from time 0.
4. Output **Yes** if and only if *satisfy = 1*.

Algorithm 2: Details of the algorithm to solve an instance of 1-FPN-3SAT1WP.

References

[AC94] S. Agarwal and A. Condon, "On Approximation Algorithms for Hierarchical MAX-SAT," *Proc of the 10th IEEE Conference on Structures in Complexity Theory,* July, 1995.

[BOW83] J.L. Bentley, T. Ottmann, P. Widmayer, "The Complexity of Manipulating Hierarchically Defined set of Intervals," *Advances in Computing Research, ed. F.P. Preparata* Vol. 1, (1983), pp. 127-158.

[CM93] E. Cohen and N. Megiddo, "Strongly Polynomial-time and NC Algorithms for Detecting Cycles in Dynamic Graphs," *Journal of the ACM (J. ACM)* Vol. 40, No. 4, September 1993, pp. 791-830.

[CM91] E. Cohen and N. Megiddo, "Recognizing Properties of Periodic graphs", *Applied Geometry and Discrete Mathematics, The Victor Klee Festschrift* Vol. 4, P. Gritzmann and B. Strumfels, eds., ACM, New York, 1991, pp. 135-146.

[CF+93] A. Condon, J. Feigenbaum, C. Lund and P. Shor, "Probabilistically Checkable Debate Systems and Approximation Algorithms for PSPACE-Hard Functions", in *Proc. 25th ACM Symposium on Theory of Computing (STOC)*, 1993, pp. 305-313.

[DP60] M.Davis and H.Putnam, "A Computing Procedure for Quantification Theory," *Journal of the ACM (J. ACM)* Vol. 7, 1960, pp. 151-158.

[FF58] L.R. Ford and D.R. Fulkerson, "Constructing Maximal Dynamic Flows from Static Flows," *Operations Research,* No. 6, 1958, pp. 419-433.

[Ga82] H. Galperin "Succinct Representation of Graphs," Ph.D. Thesis, Princeton University, 1982.

[Ga59] D. Gale, "Transient Flows in Networks," *Michigan Mathematical Journal*, No. 6, 1959 , pp. 59-63.

[GJ79] M.R. Garey and D.S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, Freeman, San Francisco CA, 1979.

[HLW92] F. Höfting, T. Lengauer and E. Wanke, "Processing of Hierarchically Defined Graphs and Graph Families," in *Data Structures and Efficient Algorithms* (Final Report on the DFG Special Joint Initiative), Springer-Verlag, LNCS 594, 1992, pp. 44-69.

[HU] J. E. Hopcroft, and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation* Addison Wesley, Reading MA., 1979.

[Hu73a] H.B. Hunt III, "On The Time Complexity of Languages," *Proc. 5th Annual ACM Symposium on Theory Of Computing, (STOC)*, 1973, pp. 10-19.

[IS87] K. Iwano and K. Steiglitz, "Testing for Cycles in Infinite Graphs with Periodic Structure," *Proc. 19th Annual ACM Symposium on Theory of Computing, (STOC)*, 1987, pp. 46-53.

[JL77] N.D. Jones and W.T. Laaser, "Complete Problems for Deterministic Polynomial Time," *Theoretical Computer Science*, No. 3, 1977, pp. 105-117.

[KMW67] R.M. Karp, R.E. Miller and S. Winograd, "The Organization of Computations for Uniform Recurrence Equations," *Journal of the ACM (J. ACM)*, Vol. 14, No. 3, 1967, pp. 563-590.

[KO91] M. Kodialam and J.B. Orlin, "Recognizing Strong Connectivity in Periodic graphs and its relation to integer programming," *Proc. 2nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1991, pp. 131-135.

[KS88] K. R. Kosaraju and G.F. Sullivan, "Detecting Cycles in Dynamic Graphs in Polynomial Time," *Proc. 27th IEEE Symposium on Foundations of Computer Science (FOCS)*, 1988, pp. 398-406.

[Le86] T. Lengauer, "Exploiting Hierarchy in VLSI Design," *Proc. AWOC '86*, Springer-Verlag, LNCS 227, 1986, pp. 180-193.

[LW87a] T. Lengauer and E. Wanke, "Efficient Solutions for Connectivity Problems for Hierarchically Defined Graphs ," *SIAM Journal on Computing*, Vol. 17, No. 6, 1988, pp. 1063-1080.

[Le89] T. Lengauer, "Hierarchical Planarity Testing," *Journal of the ACM (J. ACM)* , Vol. 36, No. 3, July 1989, pp. 474-509.

[LW92] T. Lengauer and K.W. Wagner, "The Correlation Between the Complexities of Non-Hierarchical and Hierarchical Versions of Graph Problems," *Journal of Computer and System Sciences (JCSS)*, Vol. 44, 1992, pp. 63-93.

[Li82] D. Lichtenstein, "Planar Formulae and their Uses", *SIAM Journal on Computing*, Vol 11, No. 2, May 1982 , pp. 329-343.

[MH+93a] M.V. Marathe H.B. Hunt III, and S.S. Ravi, "The Complexity of Approximating PSPACE-Complete Problems for Hierarchical Specifications," *Proc. 20th International Colloquium on Automata Languages and Programming (ICALP)*, July, 1993, pp. 76-87.

[MH+94] M.V. Marathe, H.B. Hunt III, R.E. Stearns and V. Radhakrishnan, "Approximation schemes for PSPACE-Complete problems for succinct graphs," *Proceedings of 26th Annual ACM Symposium on the Theory of Computing (STOC)*, May 1994, pp 468-477.

[Ma94] M.V. Marathe *Complexity and Approximability of NP- and PSPACE-hard Optimization Problems,* Ph.D. thesis, Department of Computer Science, University at Albany, Albany, NY August, 1994.

[MTM92] J.O. McClain, L.J. Thomas and J.B. Mazzola, *Operations Management,* Prentice Hall, Englewood Cliffs, 1992.

[MC80] C. Mead and L. Conway, *Introduction to VLSI systems* Addison Wesley, 1980.

[MS81a] B.Monien and I.H.Sudborough, "Bounding the Bandwidth of NP-Complete Problems," *Proc. 13th ACM Annual Symposium on Theory of Computing (STOC)*, 1981, pp. 279-292.

[Or82a] J.B. Orlin, "The Complexity of Dynamic/Periodic Languages and Optimization Problems," Sloan W.P. No. 1679-86 July 1985, Working paper, Alfred P. Sloan School of Management, MIT, Cambridge, MA 02139. A Preliminary version of the paper appears in *Proc. 13th ACM Annual Symposium on Theory of Computing (STOC)*, 1978, pp. 218-227.

[Or84b] J.B. Orlin, "Some Problems on Dynamic/Periodic Graphs," *Progress in Combinatorial Optimization*, Academic Press, May 1984, pp. 273-293.

[Pa94] C. Papadimitriou, *Computational Complexity* Addison-Wesley, Reading, Massachusetts, 1994.

[RH93] D.J. Rosenkrantz and H.B. Hunt III, "The Complexity of Processing Hierarchical Specifications", *SIAM Journal on Computing*, Vol. 22, No. 3, 1993, pp. 627-649.

[Sa70] W.J.Savitch, "Relationships Between Nondeterministic and Deterministic Tape Complexities," *Journal of Computer and System Sciences (JCSS)*, Vol. 4, No. 2, pp. 177-192, 1970.

[Sc78] T. Schaefer, "The Complexity of Satisfiability Problems," *Proc. 10th ACM Symposium on Theory of Computing (STOC)*, 1978, pp. 216-226.

[Wa84] K.W. Wagner, "The Complexity of Problems Concerning Graphs with Regularities", *Proc. 11th Symposium on Math. Foundations of Computer Science (MFCS)*, LNCS 176, Springer-Verlag, 1984, pp. 544-552.

[Wa93] E. Wanke, "Paths and Cycles in Finite Periodic Graphs," *Proc. 20th Symposium on Math. Foundations of Computer Science (MFCS)*, LNCS 711, Springer-Verlag, 1993, pp. 751-760.

[Wi90] M. Williams, "Efficient Processing of Hierarchical Graphs ," *TR 90-06*, Dept of Computer Science, Iowa Sate University. (Parts of the report appeared in WADS'89, pp. 563-576 and SWAT'90, pp. 320-331 coauthored with Fernandez-Baca.)

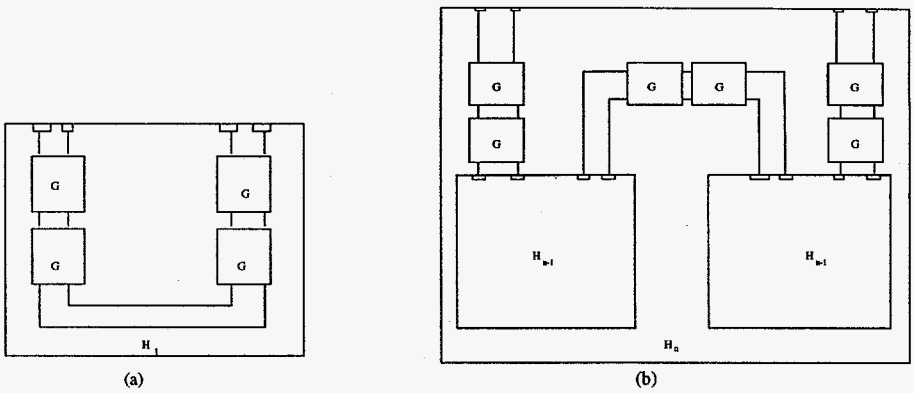

Figure 1: Construction of H_i, $1 \leq i \leq n$

Appendix

7 Translation Theorem

Theorem 7.1 There is a polynomial time transformation **TFORM**, that maps a 1-FPN(BC)-specification $\Gamma = (G(V,E), m)$ of the graph Γ^m to a strongly 1-level-restricted L-specification Γ_1 of a graph, such that the graphs represented by Γ and Γ_1 are isomorphic and $size(\Gamma_1) = (size(\Gamma))^2$.

Proof: Given a 1-FPN-specification $\Gamma = G(V,E)$ of G^m we construct a L-specification $H_G = (H_1, \cdots H_p)$ as follows. The non-terminal H_1 consists of four copies of G (i.e. G^4). They are connected in series as shown in Figure 1(a),

Non-terminal H_i, $2 \leq i \leq n$: Asssume that the periodic graph is of the form G^{2k}. The non-terminal H_i consists of five components l_i, r_i, m_i, H_{i-1}, H_{i-1}. Each of l_i, r_i and m_i consists of at least 2 copies of G (G^2) attached in a manner as shown in the Figure 1(b). Each H_{i-1} recursively encodes G^p, We need to ensure that for each recursive call p is even, thus we need to adjust the sizes of the components l_i, r_i and m_i accordingly. The H_{i-1}'s are connected to the components l_i, r_i and m_i as shown in Figure 1(b). The condition that each H_{i-1} should have even number of copies of G implies that the construction should satisfy the following constraints.

$$|l_i|,\ |r_i|,\ |m_i| \geq 2, \quad p = 2m, \quad 2m + 2m + |l_i| + |r_i| + |m_i| = 2k$$

The first constraint is needed so that the resulting specification is 1-level-restricted. The second equation says that p is even. The third equation says that the size of the components total number of copies is no more than $2k$, which is the size of the graph we want to encode. It is clar that the sizes of l_i, r_i and m_i can be chosen so that the above equations can be satisfied. It is now easy to see that the process can be carried out recursively to obtain the required specification H in polynomial time. ■

Observe that a similar transformation can be carried out if we start with a 1-FPN(BC) specified graph G. In such a case the graphs which specify the boundary conditions are included in the highest non-terminal. Also, observe that a similar transformation can be carried out to transform a 1-FPN-specification $\Gamma = (U, C(i), m)$ of a formula Γ^m to an L-specification Γ_1 that specifies a formula isomorphic to Γ^m

Corollary 7.2 There is a polynomial time, $O((size(\Gamma))^2)$ size transformation that maps a 1-FPN(BC)-specification $\Gamma = (U, C(i), m)$ of the formula Γ^m to a strongly 1-level-restricted L-specification Γ_1 in such a way that the formulas represented by Γ and Γ_1 are isomorphic.

7.1 Solvability of α-2SAT, α-3SAT1WN and α-3SAT1WP

We now discuss the polynomial time solvability of the problems α-2SAT, α-SAT(S) when every relation in S is bijunctive and finally the problems α-3SAT1WN and α-3SAT1WP. Our algorithms are based on the work of Davis and Putnam [DP60], who gave a polynomial time algorithm to solve the problem 2SAT. We first recall additional definitions from [Sc78].

The logical relation R is **bijunctive** if $R(x_1, x_2, \ldots)$ is logically equivalent to some CNF formula having at most two literals in each conjunct.

The logical relation R is **complementive** if it is closed under component-wise complement, that is, for all $(a_1, \ldots, a_m)$, if $(a_1, \ldots, a_m) \in R$, then $(1 - a_1, \ldots, 1 - a_m) \in R$. The logical relation R is **1-weakly positive** if $R(x_1, x_2, \ldots)$ is logically equivalent to some CNF formula having at most one negated variable in each conjunct, such that any clause with a negated literal has no more than 1 unnegated literal. The logical relation R is **1-weakly negative** if $R(x_1, x_2, \ldots)$ is logically equivalent to some CNF formula having at most one unnegated variable in each conjunct, such that any clause with an unnegated literal has no more than 1 negated literal.

We first review the method of Davis and Putnam[DP60] to eliminate clauses and variables. These rules can be used on any 3CNF formula and are described in Algorithm 3.

Lemma 7.3 Let Step (3) in Algorithm 3 be applied to eliminate a variable v in a 3CNF formula F to obtain a new formula F'. Then, F' can be converted into an equivalent 3CNF formula in polynomial time if one of the following conditions hold:
(a) the formula F only contains 2 literal clauses.
(b) F is 1-weakly positive.
(c) F is 1-weakly negative.

Proof: Given the 3CNF formula F with the variable v, collect the clauses in which v appears negated, collect the clauses in which v appears unnegated and collect the clauses in which v does not appear. Factor v from the set of clauses in which v appears unnegated and express this set as $v \wedge A$ where A is also a CNF formula. Factor $\overline{v}$ from the set of clauses in which v appears unnegated and express this set as $\overline{v} \wedge B$ where B is also a CNF formula. The set of clauses in which v does not appear is a CNF formula R. Thus $F = (A \vee v) \wedge (B \vee \overline{v}) \wedge R$, where A, B, and R are CNF formulae which are free of v. Then F is satisfiable iff $F' = (A \vee B) \wedge R$ is satisfiable.

We now consider three cases in which F' can be expressed as an equivalent 3CNF formula in polynomial time.

Case 1: F is a 2CNF formula.
In this case, A and B are CNF formulas in which each clause has 1 literal. Let $A = u_1 \wedge u_2 \wedge \ldots \wedge u_k$ and $B = v_1 \wedge v_2 \wedge \ldots \wedge v_l$ where u_i and v_i are literals. Then $(A \vee B)$ can be expressed as a 2CNF formula with all clauses of the form $(u_i \vee v_j), 1 \leq i \leq k, 1 \leq j \leq l$. Hence F' can also be expressed as a 2CNF formula.

Case 2: F is a 1-weakly positive formula.
In this case, A is a CNF formula in which each clause either has at most 2 unnegated literals or 1 negated literal. Similarly, B is a CNF formula in which each clause has either has at most 1 unnegated literal and has no negated literals. Let $A = c_1 \wedge c_2 \wedge \ldots \wedge c_k$ and $B = c'_1 \wedge c'_2 \wedge \ldots \wedge c'_l$ where each c_i is of the form $(u_i \vee u'_i)$ or $\overline{u_i}$, and each c'_i is of the form v_i. Here u_i, u'_i and v_i are variables. Then $(A \vee B)$ can be expressed as a 1-weakly positive 3CNF formula with all clauses of either of the form $(\overline{u_i} \vee v_j)$ or of the form $(u_i \vee u'_i \vee v_j)$ for $1 \leq i \leq k, 1 \leq j \leq l$. Hence F' can also be expressed as a 1-weakly positive 3CNF formula.

Case 3: F is a 1-weakly negative formula.
This case can be handled similar to Case 2 except that each clause in A has either at most 1 negated

Procedure Clause-Elimination
Input: *A 3CNF formula F such that one of the conditions in Lemma 7.3 hold.*
Output: **Yes** *if and only if F is satisfiable.*

1. Set *flag = false* and *satisfy = 0*
2. **While** F is empty or *flag = done* **do**

(a) **If** F contains one literal clauses **then do**

i. **If** a formula F in CNF contains a variable v as a one literal clause and also contains $\overline{v}$ as a one literal clause, then F is unsatisfiable. Set *flag = done.*

ii. If case (a) does not apply, and a variable v appears as a clause in a CNF formula, then one may modify F by deleting all clauses that contain v unnegated, and deleting all occurrences of $\overline{v}$ from the remaining clauses, to obtain a new formula F_1 which is satisfiable if and only if F is satisfiable.

iii. Set $F = F_1$. **If** F is empty then set *flag = done*; *satisfy = 1.*

iv. If case (a) does not apply, and $\overline{v}$ appears as a clause in a CNF formula, then one may modify F by deleting all clauses that contain $\overline{v}$ unnegated, and deleting all occurrences of v from the remaining clauses, to obtain a new formula which is satisfiable if and only if F is satisfiable.

v. Set $F = F_1$. **If** F is empty then set *flag = done*; *satisfy = 1.*

(b) If a variable v occurs only unnegated in a formula F or if v occurs only negated, then all clauses which contain v may be deleted. The new formula obtained is satisfiable if and only if F is satisfiable. If the new formula is empty, then F is satisfiable.

(c) Eliminating variables:

i. Let A_1 denote the formula which is a conjunction of clauses in F which contain v. Then A is obtained from A_1 by deleting v from each clause in A_1. Similarly, B be the formula which is obtained after deleting from each clause in B_1 the occurrence of $\overline{v}$.

ii. Let R be a conjunction of clauses which neither contain v nor $\overline{v}$. Then the original formula F can be put in the form: $(A \vee v) \wedge (B \vee \overline{v}) \wedge R$.
Remark: F is satisfiable if and only if $(A \vee B) \wedge R$ is satisfiable.

iii. Convert $(A \vee B) \wedge R$ into an equivalent 3CNF formula F_1 as outlined in Lemma 7.3.
Remark: The formula $(A \vee B) \wedge R$ can be converted back into 3CNF by using the laws of distributivity if the formula F only contains 2 literal clauses or if F is 1-weakly positive or if F is 1-weakly negative as shown in Lemma 7.3.

iv. Set $F = F_1$.

3. **If** *satisfy = 1* **then return Yes**
else return no.

Algorithm 3: Details of the Davis Putnam algorithm to eliminate variables.

literals and no unnegated literals, and each clause in B has at most 2 negated literals or at most 1 unnegated literal. ■

Solving α-2SAT and is variants

We now prove the polynomial time solvability of the problems L-2SATand L-SAT$_c$(S), when every relation in S is bijunctive. The procedure is described in Algorithm 4.

Proof of Correctness

Consider an instance of L-2SAT. Let the formula be of the form $F = (F_1, ..., F_n)$ where each F_i is a formula consisting of calls to $F_j (1 \leq j < i)$ and a 2CNF formula f_i. For each F_i, the 2CNF formula F_i^b, obtained in Step 1(c) of the Algorithm 4 has the following properties.

1. F_i is satisfiable if and only if F_i^b is satisfiable.
2. F_i' which is a 2CNF formula which depends only on its input variables X^i.

Heuristic ALG-L2SAT
Input: *An L-specification* $F = (F_1, \ldots, F_n)$ *of a 2SAT formula* $E(F)$.
Output: **Yes** *if and only if* $E(F)$ *is satisfiable.*

1. Repeat the following steps for $1 \leq i \leq n$.

   (a) Let $F_{i_1}, \ldots, F_{i_k}$ denote the non-terminals called in F_i. Replace each of the F_{i_j} by the smaller set of equations $F_{i_j}^b$ that has been already computed.
   Remark: The sizes of F_{i_j}, $1 \leq j \leq k$ is $O(n_{i_j}^4)$.

   (b) Let $G_i(X^i)$ represent the set of 2CNF clauses over $X^i \cup Z^i$ obtained as a result of substitution.

   (c) **If** ($i < n$) **do**

   - Eliminate the variables which are not input variables to F_i using the method of Davis and Putnam to remove one literal and two literal clauses to obtain F_i^b.
     Renmark: The variables of F_i^b are the input variables of F_i and the size of F_i^b is bounded by polynomial in the number of the input variables of F_i. This is because the number of *distinct* 2CNF clauses is on n variables is $O(n_i^4)$.

   else

   - Output **Yes** if and only if the formula G_n is satisfiable.

2. Output $H = (H_1, ..., H_n)$ as the L-specification of the satisfying assignment to the variables.

Algorithm 4: Details of the algorithm to solve L-2SAT.

3. The size of F_i^b is polynomial in the size of the input specification.

Theorem 7.4 1. Given an instance $F = (F_1, \ldots, F_n)$ of the problem L-2SAT, the algorithm ALG-L-2SAT decides in polynomial time if F is satisfiable.

2. There is a polynomial time algorithm for solving the problem L-SAT(S) when every relation in S is bijunctive.

Proof:
Part (1): Follows from the Observations 1, 2 and 3 and Algorithm 4.
Part (2): If every relation in S is bijunctive, then any instance of L-SAT(S) or L-SAT$_c$(S) can be expressed as an instance of L-2SAT in polynomial time. ■

We now consider the problems α-3SAT1WN, α-3SAT1WP and those α-3SAT(S), when every relation in S is 1-weakly positive or every relation in S is 1-weakly negative. We first prove a technical lemma about representing every 1-weakly postive or 1-weakly negative relation as a 1-weakly postive or a 1-weakly negative formula respectively.

Lemma 7.5 Every 1-weakly positive relation can be expressed as an existentially quantified 3CNF formula in which each clause is 1-weakly positive. Every 1-weakly negative relation can be expressed as an existentially quantified 3CNF formula in which each clause is 1-weakly negative.

Proof: We consider the case of a 1-weakly positive relation. This relation can be expressed as a CNF formula in which each clause has at most one negated literal. Moreover, any such clause with a negated literal has no more than 1 unnegated literal. Observe that by preceeding discussion it is clear that each clause with more than 3 literals only has unnegated literals. For each such clause (with at least 3 literals), $c = (v_1 \vee v_2 \vee \ldots \vee v_k)$, we replace c with a existentially quantified formula f_c defined as

$$f_c \equiv (v_1 \vee y_1) \wedge (\overline{y_1} \vee v_2 \vee y_2) \wedge (\overline{y_2} \vee v_3 \vee y_3) \ldots \wedge (\overline{y_{k-2}} \vee v_{k-1} \vee y_{k-1}) \wedge (\overline{y_{k-1}} \vee v_k)$$

Heuristic ALG-L3SAT1WP
Input: *An L-specification $F = (F_1, \ldots, F_n)$ of a 3SAT1WP formula $E(F)$.*
Output: **Yes** *if and only if $E(F)$ is satisfiable.*

1. Repeat the following steps for $1 \leq i \leq n$.

   (a) Let $F_{i_1}, \ldots, F_{i_k}$ denote the non-terminals called in F_i. Replace each of the F_{i_j} by the smaller set of equations $F^b_{i_j}$ that has been already computed.
   Remark: The sizes of F_{i_j}, $1 \leq j \leq k$ is $O(n^4_{i_j})$. The resulting formula is a 1-weakly positive CNF formula whose size is polynomial in the original representation of F_i.

   (b) Let $G_i(X^i)$ represent the set of clauses over $X^i \cup Z^i$ obtained as a result of substitution.

   (c) **If** ($i < n$) **do**

   - Eliminate the variables which are not input variables to F_i using the method given in Algorithm 3. The 1-weakly positive CNF formula obtained after this step is F^b_i.
   Remark: The variables of F^b_i are the input variables of F_i and the size of F^b_i is polynomial in the number of the input variables of F_i.

   else

   - Output **Yes** if and only if the formula G_n is satisfiable.

2. Output $H = (H_1, ..., H_n)$ as the L-specification of the satisfying assigment to the variables.

Algorithm 5: Details of the algorithm to solve L-3SAT1WP.

such that the y_i are existentially quantified. By direct inspection, we get that given an assignment to the variables v_i, $1 \leq i \leq k$, this assignment satisfies c if and only if the assignment can be extended to an assignment to the existentially quantified variables y_i which satisfy the formula f.

A 1-weakly negative 3CNF formula can be obtained in a similar fashion from a 1-weakly negative relation. ■

Theorem 7.6 1 The problems L-3SAT1WN and L-3SAT1WP are in P.

2 The problems L-SAT(S) and L-SAT$_c$(S) are in P if every relation in S is 1-weakly positive or every relation in S is 1-weakly negative.

Proof Sketch:
Part (1): It is easy to see that Algorithm 5 indeed decides in polynomial time whether an instance of L-3SAT1WP is satisfiable. This is because we are merely mimicking the algorithm outlined for the same problem when instances are specified using standard specifications.
Part (2): If every relation in S is 1-weakly positive, then any instance of L-SAT(S) or L-SAT$_c$(S) can be expressed as an instance of L-3SAT1WP using Lemma 7.5. Similarly, if every relation in S is 1-weakly negative, then any instance of L-SAT(S) or L-SAT$_c$(S) can be expressed as an instance of L-3SAT1WN using Lemma 7.5. ■

8 Applications

8.1 Planar Satisfiability and Graph Problems

We first show that the problem 1-FPN-PI3SAT is PSPACE-complete. To do this, we introduce some additional notation. By the proof of Theorem 5.1, we know that the instance of 1-FPN-3SAT instance has the following property: A clause defined at time t is is made up of literals defined at times t and

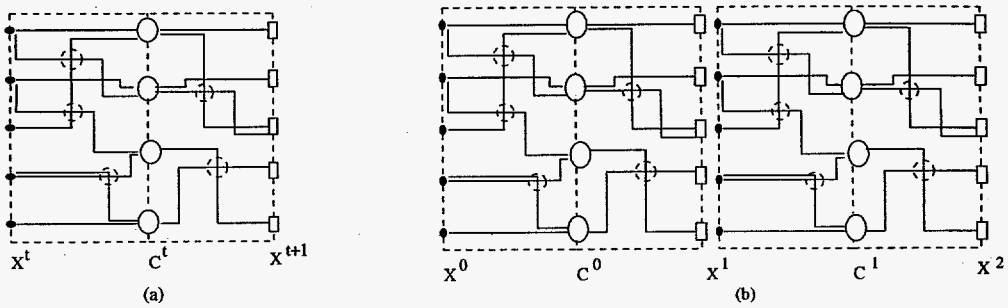

Figure 2: Schematic Diagram showing how to obtain an instance of 1-FPN-PI3SAT. (a) Schematic diagram of the bipartite static graph. The black dots represent the variables and the ellipses represent the clauses. The dotted circles denote the crossovers which are replaced by crossover boxes. The squares denote the placeholders for variables at the next time unit. The variables and clauses are knitted as shown by a dotted cycle. (b) Part of the expanded 1-FPN-PI3SAT instance. The small squares in time t are replaced by the corresponding actual variables (represented by black dots) defined in next time unit. We have left the squares and black dots representing the variables to illustrate the way they will be glued.

$t+1$. Now, given an instance of F of 1-FPN-3SAT (i.e. the static formula), we define a **extended static bipartite graph** $G(V,E)$ associated with F as follows: $V = U(t) \cup U(t+1) \cup C(t)$, where $U(t)$ corresponds to the set of variables defined at time t, $C(t)$ corresponds to the set of clauses defined at time t and $U(t+1)$ corresponds to the set of variables defined at time $(t+1)$.
$E\{(c,a)|c \in C(t), \quad a \in U(t) \cup U(t+1)$ and a appears in the clause $c\}$. One can think of $U(t+1)$ as a set of place holders for the variables defined in the next time unit. Hence the essential difference between the standard static graph and the one defined here is that instead of having weights on the edges corresponding to the period of the edge, we have two copies of the variable vertices in each period.

Now, given an instance (F,m) of 1-FPN-3SAT, we obtain an instance (F_1,m) of 1-FPN-PI3SAT as follows.

Step 1: Construct an extended static bipartite graph G associated with F.

Step 2: Next, lay out the vertices $U(t)$ $U(t+1)$ and $C(t)$ such that the following conditions hold:
(a) The vertices in $U(t)$ all lie on a distinct vertical line and are at integral coordinates. Similarly, the vertices in $U(t+1)$ and $C(t)$ all lie on a vertical line and are at integral coordinates. The relative positions of these vertical lines on which the variables in $U(t)$ $U(t+1)$ and $C(t)$ is as depicted in Figure 2.

(b) Draw the edges between $U(t)$ and $C(t)$ so that they are made up of horizontal and vertical segments and all the edges lie inside the cycle (shown in dotted lines) knitting the variables in $U(t)$ and the clauses in $C(t)$. A similar procedure is carried out for edges between $U(t+1)$ and $C(t)$.

Step 3: Remove all the crossovers (marked in the Figure 2 by dotted circles) by using Lichtenstein's crossoverbox. (Observe that the removal of with at most one negated literal per clause static bipartite grap static bipartite graph.)

The resulting formula F_1 has the property that the F_1^m is satisfiable if and only if F^m is satisfiable. The construction gives rise to a new bipartite graph G_1 which can be obtained from G in polynomial time. Moreover the graph G_1^m (obtained by expanding G_1 as shown in Figure 2(b) is planar. This is because the ordering of variables in $U(t)$ and $U(t+1)$ is consistent and hence when we identify the set of variables $U(t+1)$ with the set of variables U in the next time unit, the ordering does not change. This completes the proof of the theorem. ■