INTEGERS: ELECTRONIC JOURNAL OF COMBINATORIAL NUMBER THEORY 7 (2007), #G08

SOME HARDNESS RESULTS FOR QUESTION/ANSWER GAMES

Sarmad Abbasi
Department of Computer Science, National University of Computer and Emerging Sciences, Block B, Faisal Town, Lahore, Pakistan
sarmad.abbasi@nu.edu.pk

Numan Sheikh
Department of Computer Science, Lahore University of Management Sciences, Opposite Sector "U", DHA, Lahore, Pakistan
numan@lums.edu.pk

Received: 11/15/06, Revised: 7/9/07, Accepted: 8/8/07, Published: 9/4/07

Abstract

Question/Answer games (Q/A games) are a generalization of Ulam's game and they model information extraction in parallel. A Q/A game, $G = (D, s, (q_1, \ldots, q_r))$, is played on a directed acyclic graph, $D = (V, E)$, with a distinguished start vertex s, between two players, Paul and Carole. In the i-th round, Paul selects a set, $Q_i \subseteq V$, of at most q_i non-terminal vertices. Carole responds by choosing an outgoing edge from each vertex in Q_i. At the end of r rounds, Paul wins if and only if Carole's answers define a unique path from the root to one of the terminal vertices in D.

We explore the complexity of determining if Carole wins a Q/A game G. We show that the problem is **NP**-hard if the game is restricted to one question per round, except for the last round. The problem remains **NP**-hard if we restrict the game to two rounds. The general version of the game is **PSPACE**-complete.

1. Introduction and Definitions

Q/A games are motivated by the famous *Twenty Questions* and can be considered as a generalization of Ulam's game. A Q/A game is a *perfect information game* that is played between two persons, Paul and Carole.[1] It models how an algorithm can extract information

[1]Joel Spencer [13] has suggested that the players in these types of search games be called Paul and Carole: Paul represents the great questioner Paul Erdős; whereas, Carole, being an anagram of "oracle", represents the notoriously obtuse oracle of Apollo at Delphi.

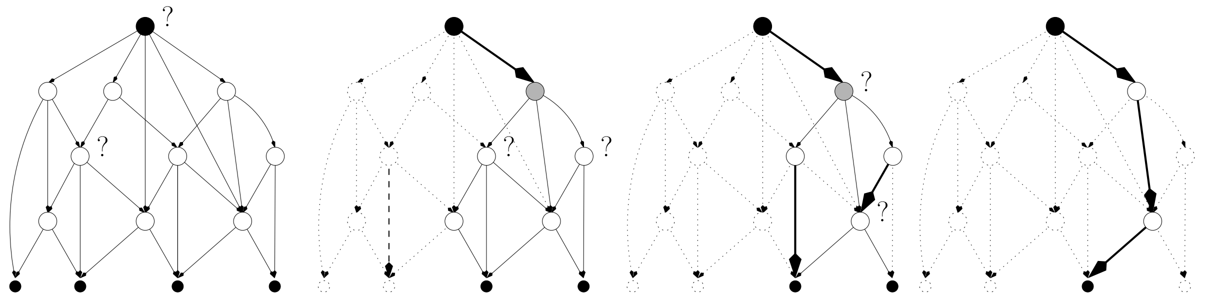

Figure 1: A Q/A game: The grey vertex is the pseudo-root. The little black vertices are the reachable terminal vertices. "?" represent Paul's questions. Bold arrows represent the choices made by Carole (Note that Carole wastes a question of Paul in the first round). Unreachable part of the graph is dotted out.

by probing the input in parallel.

Formally, a Q/A game, $G = (D, s, (q_1, \ldots, q_r))$, where:

1. $D = (V, E)$ is a directed acyclic graph.
2. The vertex $s \in V$ is a distinguished vertex called the *root*.
3. There are r rounds in the game and in the i-th round Paul is allowed to ask at most q_i questions.

We will refer to $\mathbf{q} = (q_1, \ldots, q_r)$ as the *question vector*. We will also assume that $q_i > 0$ for all i since it does not make sense to have a round with zero questions. If the maximum number of questions in each round is same; that is, $q_1 = q_2 = \ldots = q_r = q$, we denote the game by $G = (D, s, q, r)$. Furthermore, when the root of D is clear from the context we may ignore specifying it.

Most of the graph theoretic terminology is borrowed from Bollobás' monograph [3]. For a vertex v, we let $N^+(v)$ denote the set of all vertices w such that (v, w) is an edge in D. $d^+(v) = |N^+(v)|$ is the *out-degree* of the vertex v. $I = \{v : d^+(v) > 0\}$ denotes the set of *internal vertices* of D, and $T = V \setminus I$ denotes the set of *terminal vertices* of D.

Throughout this section we assume that $G = (D, s, (q_1, \ldots, q_r))$ is the game under consideration and I and T are the internal and terminal vertices of D, respectively.

In the i-th round, Paul chooses a set $Q_i \subseteq I$, such that $|Q_i| \le q_i$. If $v \in Q_i$ we say that Paul *inquires* or *asks* about the vertex v. We may also refer to v as a "question" posed in the i-th round. Carole *replies* by declaring the value of $f_i(v) \in N^+(v)$ for all $v \in Q_i$. When Carole declares $f_i(v)$ we say that she *responds* by $f_i(v)$ or Carole *points v to* $f_i(v)$. In this case, we may also say that Carole *chooses* the edge $(v, f_i(v))$.

Let U_i denote the set of questions posed by Paul till the end of the i-th round; that is, $U_i = \bigcup_{j=1}^{i} Q_i$. After the end of the i-th round, the pair (U_i, f_i), where $f_i : U_i \mapsto V$ and $f_i(v) \in N^+(v)$, completely determines the *state* or *position* of the game. Let $P = v_0, \ldots, v_t$ be a path in D. We say that P is *consistent* with the position (U_i, f_i) if $v_j \in U_i \Rightarrow v_{j+1} = f_i(v_j)$ for all $0 \leq j < t$. When the position is clear from the context we say that P is consistent with Carole's answers. To simplify the exposition we assume that Paul never repeats a question (or equivalently we can require that Carole once having answered a vertex v, does not change the value of $f_i(v)$). At the end of r rounds, Paul wins G if there is a unique path that is consistent with (U_r, f_r), the final state after r rounds of the game.

An equivalent formulation of Q/A games is given by algorithms that *probe* information in parallel. An *input* for G is a function $f : I \to V$ where $f(v) \in N^+(v)$. Note that each input naturally defines a path P_f from the root to one of the terminal vertices of D. The following theorem is easy to prove.

Theorem 1 *Paul wins $G = (D, s, \mathbf{q})$ if and only if there exists a decision tree algorithm (or strategy) that probes q_i values of f in the i-th step and at the end of k steps outputs P_f.*

For a given directed graph D, we say that the vertex t is *reachable* from the vertex s if there exists a path from s to t. Note that, if initially there is a vertex v which is not reachable from s, then Paul, playing perfectly, will never inquire about this vertex. Similarly, after the first round, if there is a vertex v such that there are is no path from s to v that is consistent with Carole's answers then Paul will not inquire about v in the subsequent rounds. Consider G in position (U_i, f_i) after i rounds. We call a vertex v *reachable with respect to the position* (U_i, f_i) if the path from the root to v is consistent with Carole's answers.

In the rest of the paper, when the position (U_i, f_i) is clear from the context, we will say that a vertex is reachable instead of specifically saying that it is reachable with respect to the position (U_i, f_i).

Let R_0 be the set of reachable vertices from s before the first round. Let R_i be the set of reachable vertices with respect to the position (U_i, f_i). Note that $R_i \subseteq R_{i-1}$ for $i = 1, \ldots, r$. Q/A-games can be interpreted as follows: In each round Paul presents Carole with a set Q_i of internal vertices of D with $|Q_i| \leq q_i$. Carole deletes all but one outgoing edge from each vertex $v \in Q_i$. Let $T_i = R_i \cap T$ be the set of terminal vertices that are reachable with respect to the position (U_i, f_i) in round i. Paul wants to play the game in such a way that at the end of the game T_r consists of a singleton set and Carole is trying to avoid this.

Example 1 Let us consider any game that is played on the a directed path P_n (see Figure 2). Since all internal vertices of P_n have exactly one edge coming out of them, Paul wins this game without asking any questions.

Example 1 shows that if an internal vertex v has out-degree one then Paul will not inquire about it at all. Indeed, we made sure that Paul inquires about internal vertices in Q_n (when Carole plays perfectly) by giving each internal vertex out-degree at least two.

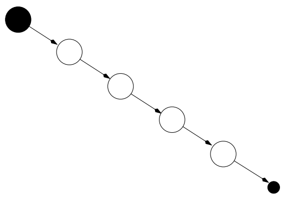

Figure 2: The Path P_5 in Example 1

Example 2 Let us consider any game that is played on the graph Q_n shown in Figure 3. This graph consist of n internal vertices connected as a path. Each one of these vertices v_i is also connected to a terminal vertex w_i. The last internal vertex is connected to two terminal vertices w_n and w_{n+1}. Paul will win the game $G = (Q_n, (q_1, \dots, q_r))$ if and only if

$$\sum_{i=1}^{r} q_i \geq n.$$

To see this note that if the above inequality is satisfied then Paul can inquire about all the internal vertices and win the game (Figure 3(c)). On the other hand if the above inequality is not satisfied, then Carole can use the following strategy and win the game. If Paul inquires about v_i with $i < n$ then she points to v_{i+1}. If he inquires about v_n then she point to w_{n+1}. Since,

$$\sum_{i=1}^{r} q_i < n$$

there is at least one internal vertex that Paul has not inquired about. Let k be the smallest index such that Paul has not inquired about v_k. It is now readily seen that

$$v_1, \dots, v_k, w_k$$

and

$$v_1, \dots, v_k, v_{k+1}, \dots, v_n, w_{n+1}$$

are two distinct paths that are consistent with Carole's answers and she wins.

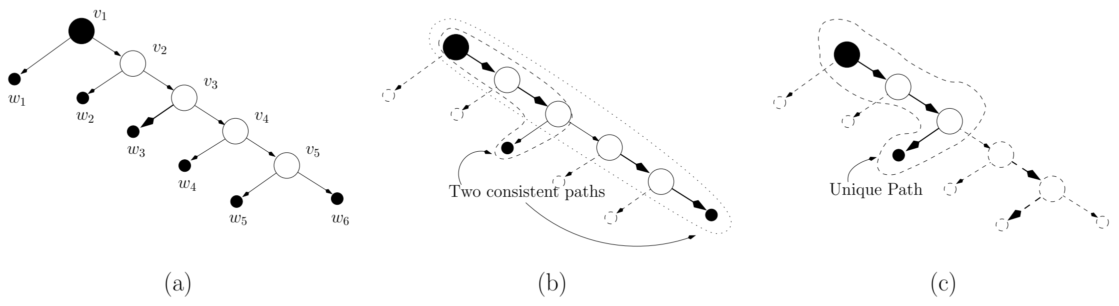

Figure 3: The Graph Q_5 in Example 2

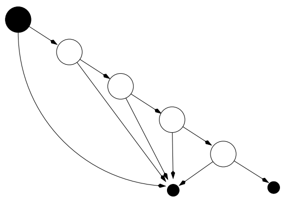

Figure 4: The Graph $\hat{Q}_5$ in Example 3

Example 3 Let $\hat{Q}_n$ be the graph in which the vertices $\{w_1, \ldots, w_n\}$ are merged into a single vertex w, as shown in Figure 4. We invite the reader to modify the argument given in Example 1.2 to show that Paul wins $G = (\hat{Q}_n, (q_1, \ldots, q_r))$ if and only if

$$\sum_{i=1}^{r} q_i \geq n.$$

We will use the digraphs like Q_n as "gadgets" in the subsequent sections. It is possible to use $\hat{Q}_n$ thereby saving some terminal vertices. However, digraphs like Q_n are clearer and allow us to make less cluttered figures.

Paul, playing perfectly, will never inquire about any internal vertex of out-degree one. This motivates the following definition. An internal vertex, v, is called *open* (with respect to (U_i, f_i)) if it is reachable and Paul has not inquired about v and the out-degree of v is at least two. A vertex x is called the *pseudo-root* (with respect to (U_i, f_i)) if x is open and all predecessors of x are not open. Once again whenever the position is clear from the context we will ignore specifying it.

Fact 2 *Paul wins $G = (D, s, (q_1, \ldots, q_r))$ if and only if the number of open vertices in the last round is at most q_r.*

It is interesting to contrast Q/A games with the r round version of Ulam's searching game [16]. In Ulam's game, $U(n; (q_1, \ldots, q_r))$, Carole thinks of an "x" from the set $S = \{1, \ldots, n\}$. Paul tries to find this x by asking questions of the form: Q_A: "Is $x \in A$?" where A can be any subset of $\{1, \ldots, n\}$. The game proceeds in r rounds and in i-th round Paul is allowed to ask q_i questions. After r rounds Paul wins if he can determine x. The outcome of this game depends only on the total number of questions asked; that is $T = \sum_{i=1}^{r} q_i$. One can easily show that Paul wins $U(n; (q_1, \ldots, q_r))$ if and only if $2^T \geq n$.

Ulam's game becomes much more interesting if Carole is allowed to lie at times [11, 12, 13]. Let $U(n, q, k)$ denote Ulam's game where Paul asks q questions adaptively; that is, he poses the i-th question after receiving the answer to the $(i-1)$-st question. Carole is allowed to lie k times. Furthermore, let $\hat{U}(n, q, k)$ be the game in which Paul asks questions a total number of q questions in two batches and Carole is allowed to lie k times. Dumitriu and Spencer

[5, 6] study $A_k(q)$, the maximal n for which Paul wins $U(n,q,k)$; and $\hat{A}_k(q)$, the maximal n for which Paul wins $\hat{U}(n,q,k)$ on an arbitrary channel (see [5, 6] for the definition of a channel). They show that $A_k(q)$ is asymptotically the same as $\hat{A}_k(q)$ (here the asymptotics are taken for a fixed k as $q \to \infty$). This feature of Ulam's game is indeed desirable since Ulam's game has connections to error-correcting codes where adaptability is undesirable. It would be extremely interesting indeed to find out the asymptotic when Paul asks all the questions in one batch.

Ulam's game is often compared [13] with the classical *Twenty Questions*. Note that few would be willing to play *Twenty Questions* if they were required to ask all of the twenty questions at once! What makes *Twenty Questions* interesting is that "answers do help in posing questions." Ulam's game does not capture this interesting aspect of *Twenty Questions*. Q/A games are an attempt to model games that do capture this interesting aspect.

Since G is a perfect information game, either Paul or Carole have a winning strategy. In that case we say G is a *Paul-win* or *Carole-win* game, respectively.

Q/A games on n-level, complete binary trees, T_n, have been analyzed in [1, 2], where it was shown that

Theorem 3 ([1, 2]) *$(T_n, (q_1, \ldots, q_r))$ is Paul-win if and only if*

$$\sum_{i=1}^{r} \lfloor \log(q_i+1) \rfloor \geq n.$$

One of the techniques used in analyzing trees is termed as *generous play* by Carole. Suppose that we allow Carole to answer any number of questions without Paul inquiring about them. Thus in any round i in the game, she answers all the questions posed by Paul, and a few more which he has not inquired about. In this case, we will say that Carole answers *generously*. It is readily seen that Paul wins a game, G, if and only if he wins G with generous play allowed to Carole. As pointed out in [2], Carole's generosity is never intended towards Paul, it is only a book keeping tool that assists in analyzing the game.

The purpose of this paper is to analyze the complexity of these games on arbitrary digraphs. Let us define the following languages:

$$\begin{aligned} \mathcal{C} &= \{\langle G \rangle : G \text{ is a Carole-win game}\} \\ \hat{\mathcal{C}} &= \{\langle G \rangle : G = (D, s, (1,1,\ldots,1,L)) \text{ is a Carole-win game}\} \\ \mathcal{C}_2 &= \{\langle G \rangle : G \text{ is a two round Carole-win game}\} \end{aligned}$$

where $\langle G \rangle$ denotes a suitable "encoding" of the game G, and L denotes and arbitrary integer.

Many combinatorial games are known to be **PSPACE**-complete [7, 9]. In general, analyzing Q/A games seems to be difficult also. We show that $\mathcal{C}$ is **PSPACE**-complete and both $\hat{\mathcal{C}}$ and $\mathcal{C}_2$ are **NP**-hard.

The rest of the paper is organized as follows: In the next section we give an overview of complexity theoretic terms that are used in our paper; this section can be easily skipped by a reader who is familiar with complexity theory. In Section 3 we prove that $\hat{\mathcal{C}}$ is **NP**-hard. In Section 4 we prove that $\mathcal{C}_2$ is **NP**-hard. In the last section we present a proof that $\mathcal{C}$ is **PSPACE**-complete.

2. Overview of Complexity Theory

In this section, we give some definitions from complexity theory that will be used in this paper. Our overview is very basic and only introduces the concepts we require. For a more thorough, general and rigorous treatment see [8, 10]. The central thesis in complexity theory asserts that the intuitive notion of algorithms is captured by the mathematically precise definition of a Turing machine. For the purpose of this paper we will not deal with Turing machines and instead only talk about algorithms.

Let $\Sigma = \{0,1\}$ be the binary alphabet. Σ^* is the set of all (finite) strings over Σ. Intuitively Σ^* is the set of all possible inputs. A *language* L over Σ is a subset of Σ^*. Let A be an algorithm and let $A(x)$ denote the output produced by A on the input x. An algorithm *decides* a language L if for every $x \in \Sigma^*$,

$$A(x) = \begin{cases} 1, & \text{if } x \in L; \\ 0, & \text{if } x \notin L. \end{cases}$$

We say that A runs in *polynomial time* if there is a k such that for every input string x the algorithm halts in time $O(|x|^k)$, where $|x|$ denotes the length of x. The class **P** is the set of all languages that can be decided by some polynomial time algorithm.

A verification algorithm V takes as input a pair (x, y). We say that V *verifies* a language L if

$$\begin{aligned} x \in L \quad &\Rightarrow \quad V(x,y) = 1 \text{ for some } y \in \Sigma^*, \text{ and} \\ x \notin L \quad &\Rightarrow \quad V(x,y) = 0 \text{ for all } y \in \Sigma^*. \end{aligned}$$

We say that V verifies L in polynomial time if there exists an integer k such that on input (x, y) the verifier V runs in time $O(|x|^k)$. It is important to note that V is not allowed to take time polynomial in the length of its input (x, y); it must run in time polynomial in the length of x only. The class **NP** is the set of all languages that can be verified in polynomial time.

The class **PSPACE** is the set of all languages that can be decided by an algorithm that requires polynomial space. The following inclusions are known (see [10]):

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}.$$

It is widely suspected that the above inclusions are strict.

Let $f : \Sigma^* \to \Sigma^*$. We say an algorithm T computes f if on input x, it produces $f(x)$ as an output. A **LOGSPACE**-transducer for $f : \Sigma^* \to \Sigma^*$ is an algorithm T that computes f and only uses $O(\log |x|)$ workspace. In this case, the algorithm is allowed to read its input several times. The output is written on a write only output device. If f can be computed by a **LOGSPACE**-transducer we say that f is **LOGSPACE**-computable.

A language A is **LOGSPACE**-reducible to B, written as $A \leq_L B$, if there exists a **LOGSPACE**-computable function f such that

$$x \in A \text{ if and only if } f(x) \in B.$$

The relation $\leq_L$ is reflexive and transitive [10]. A language L is called **NP**-hard (resp. **PSPACE**-hard) if for all languages $L' \in \mathbf{NP}$ (resp. $L' \in \mathbf{PSPACE}$), $L' \leq_L L$. A language L is called **NP**-complete (resp. **PSPACE**-complete) if $L \in \mathbf{NP}$ (resp. $L \in \mathbf{PSPACE}$) and L is **NP**-hard (resp. **PSPACE**-hard).

2.1 TQBF and SAT

We now discuss some basic definitions from propositional logic. We will identify the values 1 and 0 with the boolean constants **true** and **false**, respectively. Let $x_1, \ldots, x_n$ be a set of boolean variables. $\neg x_i$ or $\overline{x}_i$ denotes the negation of the variable x_i.

A mapping $t : \{x_1, \ldots, x_n\} \to \{0, 1\}$ is called an assignment of the variables $\{x_1, \ldots, x_n\}$. If the order of the variables $x_1, \ldots, x_n$ is clear from the context then we may some times denote an assignment t with $(t(x_1), \ldots, t(x_n))$. A literal l is a variable or its negation. The literal x_i is satisfied by all assignments t with $t(x_i) = 1$ and the literal $\overline{x_i}$ is satisfied by all assignments with $t(x_i) = 0$.

A clause C is a disjunction of literals; that is, $C = l_1 \vee l_2 \ldots \vee l_t$ where each l_j is a literal. A clause is satisfied by an assignment t if t satisfies at least one literal in C.

A formula is ϕ in conjunctive normal form (CNF) is a conjunction of clauses; that is, $\phi = C_0 \wedge C_1 \ldots \wedge C_{m-1}$, where each C_i is a clause. An assignment t satisfies ϕ if it satisfies all the clauses of ϕ. In this paper we will assume that all boolean formulae are in CNF.

Let us define

$$\text{SAT} = \{\langle\phi\rangle : \phi \text{ is in CNF and } \phi \text{ is satisfiable }\}.$$

The following theorem of Cook and Levin established the existence of **NP**-complete problems and is one of the most celebrated results in complexity theory:

Theorem 4 (Cook-Levin[4, 15]) *SAT is* **NP***-complete.*

Let ϕ be a boolean formula with n variables $x_1, \ldots, x_n$. Let $\phi|_{x_i=0}$ (resp. $\phi|_{x_i=1}$) denote the formula obtained by substituting 0 (resp. 1) for the variable x_i. A *fully quantified boolean formula*[2] ψ is a formula of the form

$$Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \phi,$$

where ϕ is a boolean formula over $\{x_1, \ldots, x_n\}$ and $Q_i \in \{\forall, \exists\}$. $\forall$ (resp. $\exists$) is called the *universal* (resp. *existential*) quantifier. If $Q_i = \forall$ (resp. $Q_i = \exists$), we say that x_i is *universally quantified* (resp. *existentially quantified*) .

A fully quantified boolean formula ϕ evaluates to 0 or 1 as follows:

1. A fully quantified boolean formula with no variables is a formula over the constants 0 and 1. It evaluates to 0 or 1 by interpreting $\wedge, \vee$ and $\neg$ in the natural way.
2. If $\psi = \forall x \psi_0$ then ψ evaluates to 1 if and only if both $\psi_0|_{x=0}$ and $\psi_0|_{x=1}$ evaluate to 1.
3. If $\psi = \exists x \psi_0$ then ψ evaluates to 1 if and only if at least one of $\psi_0|_{x=0}$ or $\psi_0|_{x=1}$ evaluate to 1.

For the purpose of our paper we will assume the quantifiers in any fully quantified boolean formula start with the existential quantifier and always alternate; that is,

$$Q_i = \begin{cases} \exists, & \text{if } i \text{ is odd;} \\ \forall, & \text{if } i \text{ is even.} \end{cases}$$

We define the language TQBF as follows:

TQBF $= \{\langle\psi\rangle : \psi$ is a fully quantified boolean formula in CNF that evaluates to 1$\}$.

Theorem 5 (Stockmeyer and Meyer[14]) *TQBF is* **PSPACE**-*complete.*

3. Complexity of $\hat{\mathcal{C}}$

The game $G = (D, s, 1, r)$ is trivial. Paul wins G if and only if the length of the longest path starting from s in the directed acyclic graph D is at most r, which can be checked in polynomial time. If we allow more questions in the last round, the game becomes hard to solve and we have the following theorem.

[2]In logic ψ would be a fully quantified boolean formula in prenex normal form with the quantifier free part in CNF. However, we omit the discussion of normal forms in our paper.

Theorem 6 *$\hat{\mathcal{C}}$ is* **NP**-*Hard.*

In order to prove the theorem we will show that SAT is reducible to $\hat{\mathcal{C}}$. As SAT is NP-complete and **LOGSPACE**-reducibility is transitive this shows that $\hat{\mathcal{C}}$ is **NP**-hard. Towards this end we show that given a boolean formula ϕ in conjunctive normal form, we can compute a Q/A game G_ϕ in **LOGSPACE** such that ϕ is satisfiable if and only if Carole wins G_ϕ. Let ϕ be a boolean formula in conjunctive normal form on n variables $\{x_1, \ldots, x_n\}$. Let m be the number of clauses in ϕ; that is,

$$\phi = C_0 \wedge C_1 \wedge \cdots \wedge C_{m-1}.$$

The game G_ϕ is played on a directed graph D_ϕ and consists $n+1$ rounds.

$$G_\phi = (D_\phi, s = v_1^0, (\underbrace{1, \ldots, 1}_{n \text{ times}}, L)),$$

where $L = nm + m - 1$. In the first n rounds Paul is allowed to ask one question and in the last round he can ask $nm + m - 1$ questions.

3.1 Construction of D_ϕ

We assume that $m \geq n$ (otherwise, we can just add n clauses with each variable and its negation to ϕ). The graph D_ϕ corresponding to the game G_ϕ consists of $2n(m+1) + 3m + 2$ vertices.

For each clause C_i there is a *clause vertex* labeled c_i. Each clause vertex is connected to two terminal vertices labeled l_i and r_i. This guarantees that if c_i is open in the last round then Paul must inquire about c_i. There are three more special terminal vertices labeled p, q and z in the graph.

For each variable x_i, other than x_1, we have two *value selection vertices* v_i^0 and v_i^1 in the graph. For x_1 we have only one value selection vertex v_1^0. Each value selection vertices v_i^0 (and v_i^1 when $i > 1$) is connected to two paths $P_i(0)$ and $P_i(1)$.

$P_i(0)$ consists of m internal vertices $\overline{w}_i(0), \ldots, \overline{w}_i(m-1)$ connected as a directed path. If $\overline{x}_i$ appears in C_j then $\overline{w}_i(j)$ is connected to the clause vertex c_j. If $\overline{x}_i$ does not appear in C_j then $\overline{w}_i(j)$ is connected to z. Note that if we do not connect $\overline{w}_i(j)$ to z then its out-degree will be one and Paul, playing perfectly, will never inquire about it. Similarly, $P_i(1)$ consists of internal m vertices $w_i(0), \ldots, w_i(m-1)$ connected as a directed path. If x_i appears in C_j then $w_i(j)$ is connected to the clause vertex c_j. If x_i does not appear in C_j then $w_i(j)$ is connected to z.

The last vertices, $\overline{w}_i(m-1)$ and $w_i(m-1)$, of the paths $P_i(0)$ and $P_i(1)$ are connected to both v_{i+1}^0 and v_{i+1}^1 for $i = 1, \ldots, n-1$. The last vertices $\overline{w}_n(m-1)$ and $w_n(m-1)$ are connected to p and q respectively. Figure 5 shows the entire graph for

$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2) \wedge (x_2 \vee x_3).$$

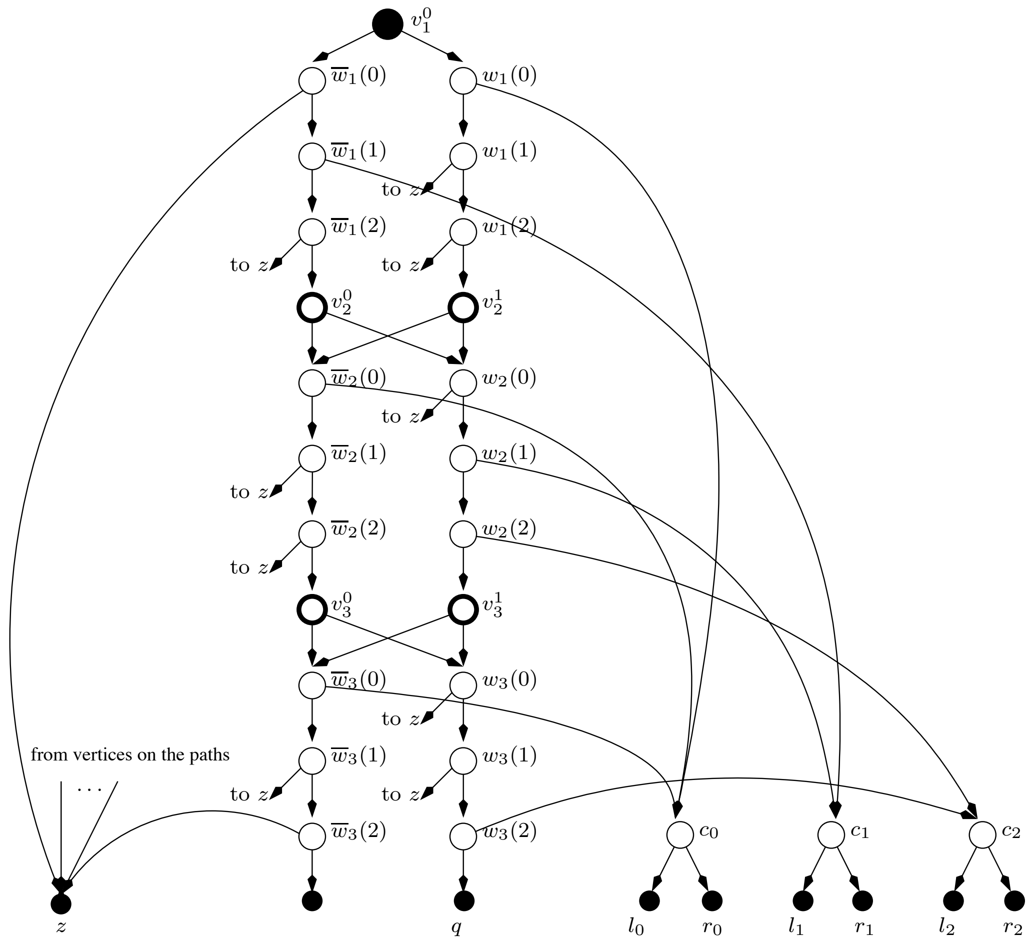

Figure 5: The graph D_ϕ. Not all the edges to the vertex z are shown.

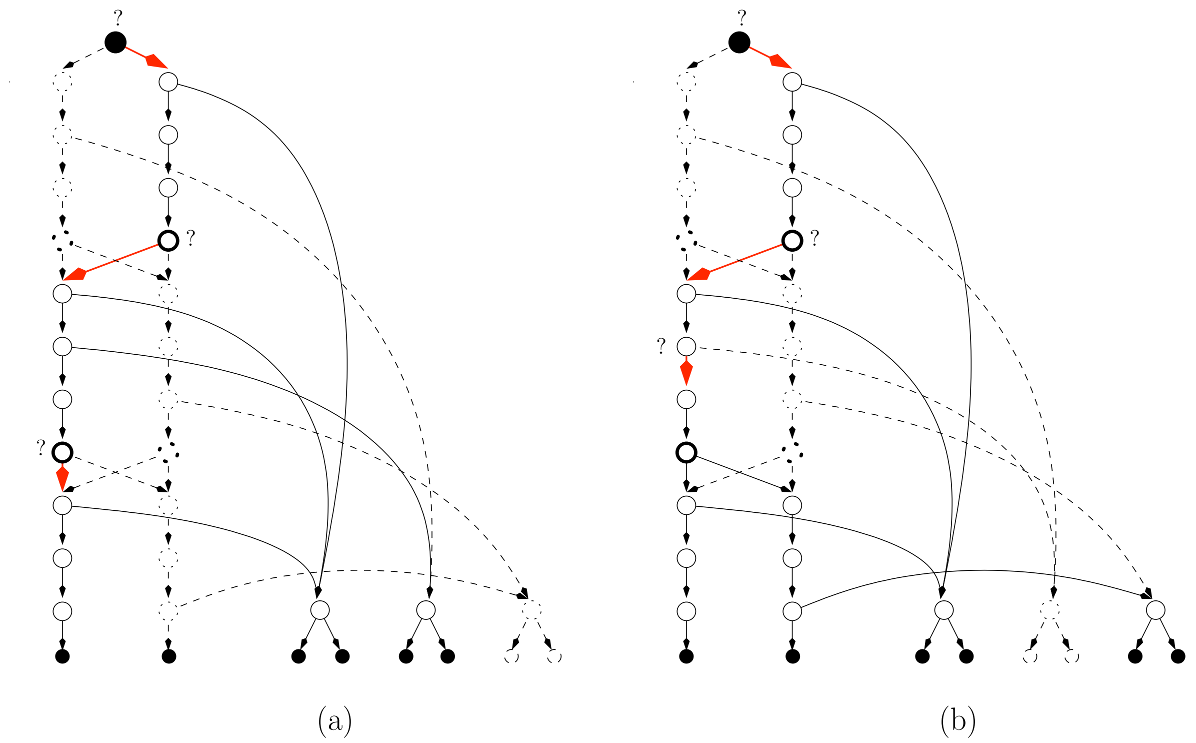

Figure 6: (a) The graph D_ϕ after normal play with assignment $(1, 0, 0)$. Unreachable part of the graph is dotted out. Note that C_2 is not satisfied by the assignment hence c_2 is not reachable. (b) The graph D_ϕ when Paul plays non-normally. He manages to cut off c_1 at a high cost. The vertex z is not shown in both the cases.

3.2 Normal Play

We first describe the *normal play* for the game G_ϕ.

Carole picks an assignment $t = (t_1, \ldots, t_n)$ of variables. In the first round, Paul asks the root v_1^0 and Carole points to the root of $P_1(t_1)$. In the i-th round (for $i = 2, \ldots, n$), Paul inquires about the value selection vertex $v_i^{t_{i-1}}$. Carole points $v_i^{t_{i-1}}$ to the root of $P_i(t_i)$.

Lemma 7 *If Paul and Carole play normally then the number of open vertices in the last round is $nm+\mu$, where μ is the number of clauses satisfied by Carole's assignment $(t_1, \ldots, t_n)$.*

Proof. Paul inquires about v_1^0 in the first round hence it is not open. At the end of normal play all the m vertices on the path $P_i(t_i)$ are open. Furthermore, all the m vertices of $P_i(1 - t_i)$ are not reachable. The vertex $v_i^{t_{i-1}}$ is reachable but inquired about and therefore not open. The vertex $v_i^{1-t_{i-1}}$ is not reachable. Hence there are nm non-clause vertices that are open.

Now, suppose that a clause C_j is satisfied by the assignment t. Then either there is a variable x_i that appears in C_j and $t_i = 1$ or there is a variable x_i whose negation appears in C_j and $t_i = 0$. In the first case, all the vertices in $P_i(1)$ are open. In particular $w_i(j)$ is open. Since there is an edge from $w_i(j)$ to c_j hence c_j is also open. In the second case, all the vertices in $P_i(0)$ are open. In particular $\overline{w}_i(j)$ is open. Since there is an edge from $\overline{w}_i(j)$ to c_j hence c_j is also open.

In case, C_j is not satisfied by the assignment t then for every variable x_i (resp. its negation $\overline{x}_i$) that appears in C_j, $t_i = 0$ (resp. $t_i = 1$). Thus all the vertices of $P_i(1 - t_i)$ are not reachable. Hence, c_j is not reachable. □

Lemma 8 *If Paul and Carole play normally then Carole wins G_ϕ if and only if ϕ is satisfiable.*

Proof. Follows from Fact 2 and Lemma 7. □

3.3 Non-normal Play

Note that if ϕ is not satisfiable then Paul can play normally and Carole has no way of winning. In the first n round her answers define a variable assignment. In the last round if there are strictly less than m clause vertices reachable, Paul has enough questions to ask all the open vertices to win the game. On the other hand if the formula is satisfiable, Paul cannot win by using normal strategy. We complete the proof of Theorem 6 by showing that if the formula is satisfiable, Paul cannot win even if he deviates from the normal strategy.

Let us assume the formula is satisfiable and Paul deviates from the normal strategy. The deviation can only come in the first n rounds, and he can deviate in the following ways in the i-th round.

Case 1 He can ask the question on a value selection vertex below v_i^0 or v_i^1. In this case, Carole would lead him to the appropriate vertex according to her satisfying assignment of the variable x_i.

Case 2 He can ask some questions on a path $P_i(0)$ or $P_i(1)$. If he inquires about $\overline{w}_i(j)$ Carole leads him to $\overline{w}_i(j+1)$ (or v_{i+1}^0 if $j = m-1$). Similarly, if he inquires about $w_i(j)$ Carole leads him to $w_i(j+1)$ (or v_{i+1}^1 if $j = m-1$). Lastly, if he inquires about $w_n(m-1)$ (resp. $\overline{w}_n(m-1)$) she leads him to p (resp. q).

Case 3 He can ask the question on a clause vertex. In this case Carole answers arbitrarily.

Let α be the number of questions asked on the value selection vertices in the first n rounds. Let β be the number of questions asked on some $P_i(b)$ where $b \in \{0, 1\}$. Lastly, let γ be the number of questions asked on the clause vertices in the first n rounds, so that

$$\alpha + \beta + \gamma = n.$$

We first deal with the case $\alpha = n$. In this case, Paul inquired about n value selection vertices. If he asks exactly one question on each of $\{v_i^0, v_i^1\}$. It is easy to see that all the vertices of $P_i(t_i)$ are reachable for each i (in fact, it is possible all the vertices of both $P_i(0)$ and $P_i(1)$ are reachable). In this case, the same analysis as one for the normal strategy shows that Paul cannot win.

If $\alpha < n$ then there is at least one i such that neither v_i^0 nor v_i^1 is inquired about. Let k be the smallest such i. In this case all the vertices in both $P_j(0)$ and $P_j(1)$ are reachable for any $j \geq k$. Hence we can count the number of open vertices after n rounds as follows:

$$\begin{aligned}
\text{no. of open value selection vertices} &\geq n - \alpha = \beta + \gamma \\
\text{no. of reachable vertices on } P_i(b) &= 2(n-k+1)m + (k-1)m \\
&= 2nm - km + m \\
\text{no. of open vertices on } P_i(b) &= 2nm - km + m - \beta \\
\text{no. of open clause vertices} &\geq m - (\beta + \gamma) \\
\text{Total no. of open vertices} &\geq \beta + \gamma + m - (\beta + \gamma) + 2nm - km + m - \beta \\
&= nm + m + (nm - km + m - \beta) \\
&\geq nm + m \text{ (because } n \geq k \text{ and } m \geq n \geq \beta) \\
&> nm + m - 1
\end{aligned}$$

This shows that Paul does not have enough question to determine the path completely.

To complete the proof of Theorem 6 we have to argue that G_ϕ can be computed from ϕ in **LOGSPACE**. Note that the vertex set of D_ϕ can be computed by counting the number of variables and clauses in ϕ which require **LOGSPACE**. The detailed algorithm that computes the edges in D_ϕ is given in Algorithm 1. We only point out that this algorithm uses a few counters and thus requires **LOGSPACE**. It is easy to see that the question vector can also be computed in **LOGSPACE**.

4. Complexity of $\mathcal{C}_2$

To prove that $\mathcal{C}_2$ is **NP**-hard, we again reduce SAT to $\mathcal{C}_2$. We use the same graph D_ϕ from the previous section. However, now the game, G_ϕ^2, has only two rounds. The first round consists of $2n-1$ questions and the second round consists of L questions, where $L = nm + m - 1$. Thus $G_\phi^2 = (D_\phi, (2n-1, L))$.

Theorem 9 *$\mathcal{C}_2$ is* **NP**-*Hard.*

In order to prove the theorem, we again describe the strategies in terms of *normal* and *non-normal* play. Note that the notion of the normal play holds only for the first round, as in the second, Paul will have to ask all the open vertices.

```
Input: $\phi$
Output: The graph, $D_\phi$
begin
    output $(v_1^0, w_1(0))$, output $(v_1^0, \overline{w}_1(0))$;
    foreach $i = 2 \ldots n$ do
        output $(v_i^0, w_i(0))$, output $(v_i^0, \overline{w}_i(0))$;
        output $(v_i^1, w_i(0))$, output $(v_i^1, \overline{w}_i(0))$;
    end
    foreach $i = 1 \ldots n$ do
        foreach $j = 0 \ldots m-2$ do
            output $(w_i(j), w_i(j+1))$, output $(\overline{w}_i(j), \overline{w}_i(j+1))$;
        end
        if $i \neq n$ then
            output $(w_i(m-1), v_{i+1}^0)$, output $(\overline{w}_i(m-1), v_{i+1}^1)$;
        else
            output $(w_n(m-1), p)$, output $(\overline{w}_n(m-1), q)$;
        end
    end
    foreach $j = 0 \ldots m-1$ do
        output $(c_i, l_i)$, output $(c_i, r_i)$;
    end
    foreach $i = 1 \ldots n$ do
        foreach $j = 0 \ldots m-1$ do
            if $x_i$ appears in $C_j$ then
                output $(w_i(j), c_j)$, output $(\overline{w}_i(j), z)$;
            else if $\overline{x}_i$ appears in $C_j$ then
                output $(\overline{w}_i(j), c_j)$, output $(w_i(j), z)$;
            else
                output $(\overline{w}_i(j), z)$, output $(w_i(j), z)$;
            end
        end
    end
end
```

Algorithm 1: **LOGSPACE** algorithm to compute the edges of D_ϕ

4.1 Normal Play

We now describe the normal play for our game G^2_ϕ for the first round. Carole picks an assignment $t = (t_1, \ldots, t_n)$ of variables. Paul asks all the $2n - 1$ value selection vertices $v^0_1, \ldots, v^0_n$ and $v^1_2, \ldots, v^1_n$. Carole responds in such a way that if $t_i = 0$ then the path $P_i(0)$ is reachable in the next round and $P_i(1)$ is unreachable in the next round and the case is reversed if $t_i = 1$. More formally,

1. Carole selects the outgoing edge from v^0_1 to $P_1(t_1)$.
2. She also selects the outgoing edge from $v^{t_{i-1}}_i$ to $P_i(t_i)$ for $i = 2, \ldots, n$.
3. She selects an outgoing edge from $v^{1-t_{i-1}}_i$ arbitrarily.

The following lemma is immediate.

Lemma 10 *In normal play the number of open vertices in the second round is*

$$nm + \textit{the number of clauses satisfied by the assignment } (t_1, \ldots, t_n).$$

In the second round, Paul has $nm+m-1$ questions. This tells us that Carole wins in normal play if and only if ϕ is satisfiable. Hence we get the following result.

Lemma 11 *If Paul plays normally then Carole wins G^2_ϕ if and only if ϕ is satisfiable.*

4.2 Non-normal Play

Paul has a winning strategy if ϕ is not satisfiable, therefore he will choose to play normally. Note that, Carole cannot deviate from normal strategy if Paul is playing normally. Answers given by Carole, to the normal strategy questions in the first round by Paul, always determine an assignment $(t_1, \ldots, t_n)$ of the variables.

On the other hand if the formula is satisfiable, Paul cannot win by normal strategy. In this section we will complete the proof of Theorem 9 by showing that if the formula is satisfiable, Paul cannot win even if he deviates from the normal strategy.

Let us assume the formula is satisfiable and Paul deviates from the normal strategy. Paul has $2n - 1$ questions in the first round. He can use the questions using non-normal play strategy as follows:

Case 1 He can ask some questions on the vertices of $P_i(0)$ or $P_i(1)$. If he inquires about $\overline{w}_i(j)$ Carole leads him to $\overline{w}_i(j + 1)$ (or v^0_{i+1} if $j = m - 1$). Similarly, if he inquires about $w_i(j)$ Carole leads him to $w_i(j + 1)$ (or v^1_{i+1} if $j = m - 1$). Lastly, if he inquires about $w_n(m - 1)$ (resp. $\overline{w_n}(m - 1)$), she leads him to p (resp. q).

Case 2 He can ask some questions on clause vertices. In this case, Carole answers arbitrarily.

Let α, β and γ be the number of questions asked on the value selection vertices, vertices on paths, and the clause vertices, respectively, in the first round. We have $\alpha+\beta+\gamma = 2n-1$. If Paul plays with non-normal strategy then $\alpha < 2n-1$. We define $k = 1$ if Paul did not inquire about v_1^0. Otherwise, there is i such that Paul inquires about at most one vertex in $\{v_i^0, v_i^1\}$. Let k be the smallest index such that one of the vertices v_k^0 and v_k^1 has not been inquired about. Carole follows the following strategy:

1. If $k > 1$ and v_k^0 is not inquired about (this includes the case when both v_k^0 and v_k^1 are not inquired about), answer v_{k-1}^0 and v_{k-1}^1 to the root of $P_{k-1}(0)$.
2. If $k > 1$ and v_k^0 is inquired about and v_k^1 is not inquired about then answer v_{k-1}^0 and v_{k-1}^1 to the root of $P_{k-1}(1)$.
3. For all $i \neq k-1$, if v_i^0 (resp. v_i^1) is inquired about then answer it to the root of $P_i(0)$ (resp. $P_i(1)$).

It is readily seen that:

1. All the vertices on the path $P_i(0)$ is reachable and none of the vertices on $P_i(1)$ are reachable $i = 1, \ldots, k-1$.
2. All the vertices on the paths $P_i(0)$ and $P_i(1)$ are reachable for $i = k, \ldots, n$.

The number of reachable vertices on paths is $(k-1)m + 2(n-k+1)m = 2nm - mk + m$. Out of these at least $2nm - mk + m - \beta$ are open in the second round. The number of value selection vertices that are open is $2n-1-\alpha = \beta+\gamma$. Hence the total number of open vertices is at least $2nm - mk + m + \gamma = L + 1 + (n-k)m + \gamma$. As $k \leq n$, this quantity exceeds L which is the number of questions allowed in the second round. Therefore, Paul cannot win by playing non-normally. We record the results of this section in the following lemma.

Lemma 12 *Carole wins G_ϕ^2 if and only if ϕ is satisfiable.* □

Again, we see that G_ϕ^2 can be computed from ϕ in **LOGSPACE**, thereby establishing Theorem 9.

5. $\mathcal{C}$ is PSPACE-complete

In this section we show that the general question of establishing the winner by looking at a Q/A game, $G = (D, s, \mathbf{q})$, is indeed very hard to answer.

Theorem 13 *$\mathcal{C}$ is* **PSPACE**-*complete.*

Let ψ be a fully quantified boolean formula,

$$\psi = \exists x_1 \forall x_2 \exists x_3 \cdots \forall x_{n-1} \exists x_n \, \phi,$$

where ϕ is the quantifier free boolean formula in CNF over n variables, $x_1, \ldots, x_n$, consisting of m clauses. We assume that the quantifiers alternate between existential and universal ones and ψ starts with an existential quantifier. Furthermore, we also assume that the last quantifier is existential; that is, n is odd (TQBF is known to be **PSPACE**-complete with these restrictions also). We describe a Q/A game, $G_\psi = (D_\psi, s, \mathbf{q})$ such that:

1. G_ψ can be computed from a quantified boolean formula ψ using **LOGSPACE**.
2. Carole wins G_ψ if and only if ψ is true.

Let $d = 15n$, $h = 15n$ and $r = 8n$ be the three other parameters that will be used to describe the game. (Note that here r is not the number of rounds in the game.) In fact, any large enough values of d, h and r will work in the construction. In order to reduce TQBF to Q/A games we can use the ideas of Theorem 6. However, we have to over come many problems. Firstly, the order in which the variables selected by Paul and Carole has to alternate. Secondly, we have to build a gadget that can perform the task of universal quantification. It is possible to make such gadgets, however, the resulting Q/A game is more elaborate then the previous ones.

5.1 Construction of D_ψ

We define the construction of digraph D_ψ on which G_ψ will be played. D_ψ contains $m \times d$ *clause vertices.* For each clause C_i there are d vertices labeled $c_i^0, \ldots, c_i^{d-1}$. Each clause vertex c_i^k is connected to two terminal vertices labeled l_i^k and r_i^k. Note that we have d copies of each clause vertex as opposed to a single copy.

For each variable, x_i, other than x_1, we create r *strands*, $S_i^1, \ldots, S_i^r$, and for x_1 we only create one strand, S_1^1. Let us define the c-th strand, S_i^c, of x_i. It is rooted at the *value selection* vertex v_i^c. The vertex v_i^c is connected to the roots of the two paths $P_i^c(0)$ and $P_i^c(1)$. The vertex v_1^1 is the root s of the graph D_ψ. In the previous proof we used just one copy of a strand instead of r copies.

$P_i^c(0)$ consists of $md+1$ vertices, namely, $\overline{w}_i^c(0), \ldots, \overline{w}_i^c(md)$. If $\overline{x}_i$ appears in clause C_j then $\overline{w}_i^c(jd+k)$ is connected to c_j^k for $k = 0, \ldots, d-1$. If $\overline{x}_i$ does not appear in clause C_j then each $\overline{w}_i^c(jd+k)$ is connected to a new terminal vertex $\overline{u}_i^c(jd+k)$ for $k = 0, \ldots, d-1$.

Similarly, $P_i^c(1)$ consists of $md+1$ vertices, namely, $w_i^c(0), \ldots, w_i^c(md)$. If x_i appears in clause C_j then $w_i^c(jd+k)$ is connected to c_j^k for $k = 0, \ldots, d-1$. If x_i does not appear in clause C_j then each $w_i^c(jd+k)$ is connected to a new terminal vertex $u_i^c(jd+k)$ for $k = 0, \ldots, d-1$.

The vertices $w_i^c(md)$ and $\overline{w}_i^c(md)$ are the *output vertices* of this strand and are connected to the value selection vertices of *all the r strands* for the next variable. The output vertices

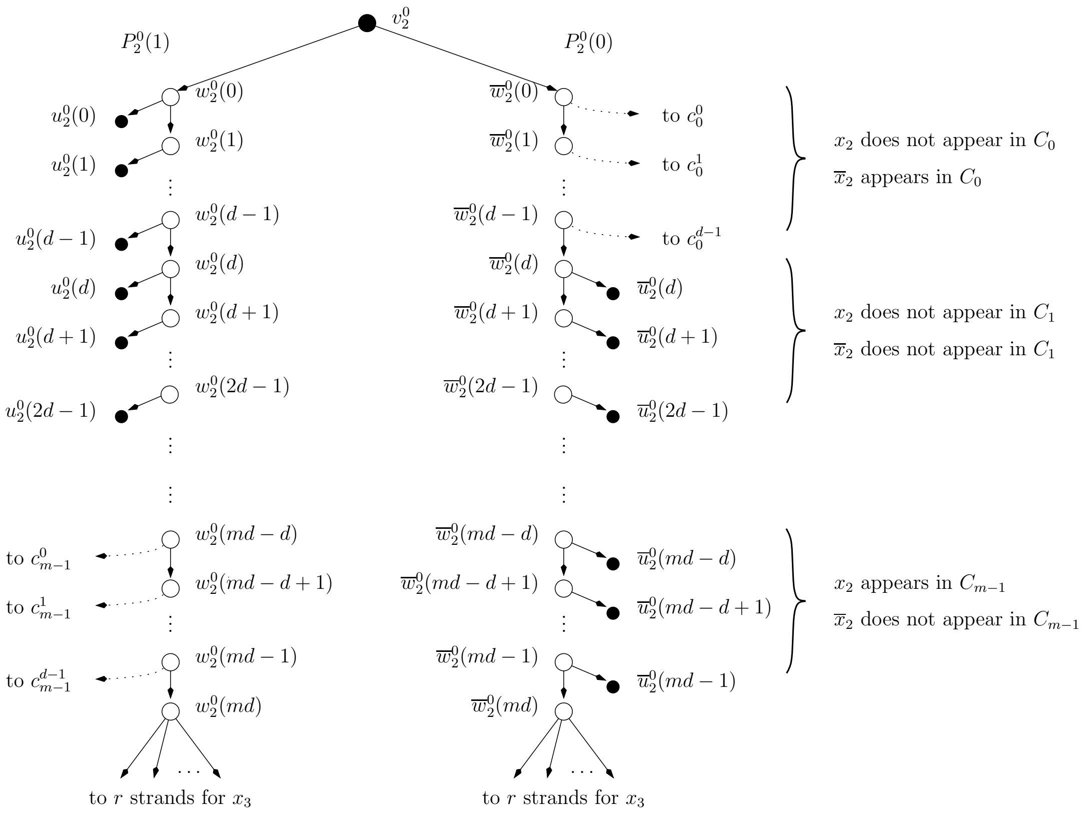

Figure 7: A strand corresponding to the variable x_2. The corresponding literal $\overline{x}_2$ appear in C_0 and x_2 appear in C_m. None of the corresponding literal appear in C_1.

of the strand S_1^1; that is, $w_1^1(md)$ and $\overline{w}_1^1(md)$, are connected to $v_2^1, \ldots, v_2^r$, which are the variable selection vertices of the strands, $S_2^1, \ldots, S_2^r$, for the second variable, x_2. Similarly, for $i = 2, \ldots, n-1$ the output vertices of the strand $S_i^1, \ldots, S_i^r$; that is, $w_i^1(md), \ldots, w_i^r(md)$ and $\overline{w}_i^1(md), \ldots, \overline{w}_i^r(md)$ are connected to $v_{i+1}^1, \ldots, v_{i+1}^r$ (Figure 7 shows a strand for x_2).

The last and one of the most important ingredients of this graph is a collection of $4r$ path like structures called the *demotivating paths*. Each one of these paths has h internal vertices connected as a path. Each internal vertex is also connected to a terminal vertex except for the last internal vertex which is connected to two terminal vertices. These paths will be used to demotivate Carole from playing non-normally. The last vertices in S_n^c; that is, $w_n^c(md)$ and $\overline{w}_n^c(md)$, for $c = 1, \ldots, r$, are connected to two demotivating paths each. The entire picture looks like Figure 8.

The game consists of $2n$ rounds. The number of questions for each round is specified by

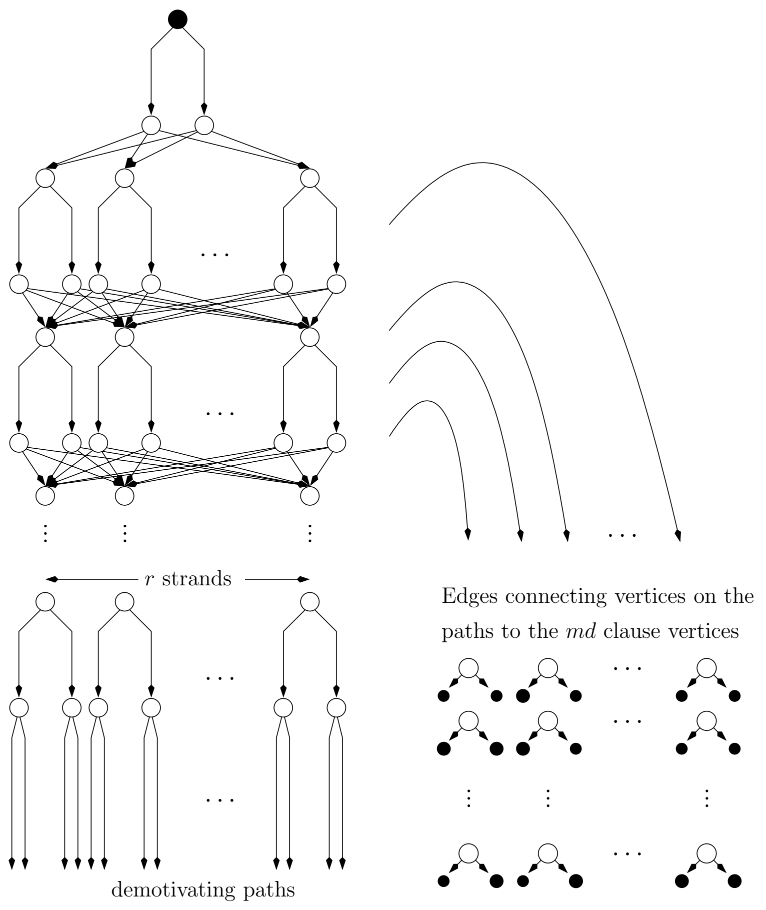
r strands
Edges connecting vertices on the paths to the md clause vertices
demotivating paths

Figure 8: Zoomed out view of the graph D_ψ.

the vector:

$$\mathbf{q} = (\underbrace{1}_{\exists x_1}, \underbrace{1, 2}_{\forall x_2}, \underbrace{1, 1}_{\exists x_3}, \underbrace{1, 2}_{\forall x_4}, \ldots, L).$$

We let

$$L = 2h + nmd + (m-1)d + 1.$$

Note that for the variable x_1 there is one round in the game. For each $1 < i \leq n$:

1. if x_i is existentially quantified then there are two rounds with one question each.
2. if x_i is universally quantified then there are two rounds with one question in the first and two in the second round.

In the last round Paul is allowed to ask L questions.

5.2 Normal Play

Let us now define *normal play* for this game. Normal play corresponds to Carole and Paul selecting the values of the variables alternately. Carole selects the values of the existentially quantified variables, whereas Paul selects the value of the universally quantified ones. This is done as follows: At the start, the current strand for variable x_1 is S_1^1. Hence, we set $c_1 = 1$. In the first round, Paul asks about v_1^1 and Carole can reply by pointing to the path $P_1^1(0)$ or $P_1^1(1)$ thereby setting the value of $x_1 = 0$ or 1 respectively. We denote the value by t_1. (At this point, the path not selected by Carole can be deleted.) $P_1^{c_1}(t_1)$ is set to be the current path. Next, Paul inquires about the last vertex of the current path. He asks $w_1^1(md)$ if $t_1 = 1$, and $\overline{w}_1^1(md)$ if $t_1 = 0$. Carole points him to the value selection vertex $v_2^{c_2}$ on a strand for the variable x_2. Now, the current strand is set to c_2. x_2 is universally quantified and Paul has two questions in this round. He uses the second question to dictate the value of x_2. Suppose Paul wants to set the value of x_2 to 0. In this case, he inquires about $v_2^{c_2}$ and the last vertex, $w_2^{c_2}(md)$, of $P_2^{c_2}(1)$. In normal play, Carole points him to $P_2^{c_2}(0)$ thereby setting $x_2 = 0$ according to his wishes. It should be noted that the last vertex $\overline{w}_2^{c_2}$ is not answered. In the next round Paul inquires about $\overline{w}_2^{c_2}$ and Carole points it to a new variable selection vertex $v_3^{c_3}$ thereby setting the current strand to c_3. Normal play continues like this till the value of all the variables have been selected. Formally, normal play is specified by the following procedure:

Normal Play:

Let $c_1 = 1$.

1. In round $2i + 1$, if i is even then x_{i+1} is existentially quantified. Paul has one question in this round and he inquires about the value selection vertex $v_{i+1}^{c_{i+1}}$. Carole points him to either $P_{i+1}^{c_{i+1}}(0)$ or $P_{i+1}^{c_{i+1}}(1)$. In this way, she sets the variable x_{i+1} to be 0 or 1 according to her choice. We denote this value by t_{i+1}.

2. In round $2i + 1$, if i is odd then x_{i+1} is universally quantified. Paul has two question and he inquires about the value selection vertex $v_{i+1}^{c_{i+1}}$ and

   (a) if he wishes to set $x_{i+1} = 0$, he inquires about the last vertex, $w_{i+1}^{c_{i+1}}(md)$, of $P_{i+1}^{c_{i+1}}(1)$. Carole points $v_{i+1}^{c_{i+1}}$ to the root of $P_{i+1}^{c_{i+1}}(0)$, setting the variable x_{i+1} to 0 according to his choice. We let $t_{i+1} = 0$. Carole points $w_{i+1}^{c_{i+1}}(md)$ to the root of some strand arbitrarily.

   (b) if he wishes to set $x_{i+1} = 1$, he inquires about the last vertex, $\overline{w}_{i+1}^{c_{i+1}}(md)$, of $P_{i+1}^{c_{i+1}}(0)$. Carole points $v_{i+1}^{c_{i+1}}$ to to the root of $P_{i+1}^{c_{i+1}}(1)$, setting the variable x_{i+1} to 1 according to his choice. We let $t_{i+1} = 1$. Carole points $\overline{w}_{i+1}^{c_{i+1}}(md)$ to the root of some strand arbitrarily.

3. In round $2i$, if $i < n$ Paul inquires about the last vertex of $P_i^{c_i}(t_i)$. Carole leads him to a value selection vertex $v_{i+1}^{c_{i+1}}$. The current strand is set to c_{i+1}.

With normal play, after $2n-1$ rounds, Paul has discarded all but one strand for each variable. Furthermore, only one path is reachable in each strand according to the values of the variables selected. Let us count the total number of open vertices in the game. There are n paths, $P_i^{c_i}(t_i)$ for $i = 1, \ldots, n$, reachable on the strands. Each one, except the last, has the last vertex answered. Hence, the total number of open vertices on these paths is $nmd + 1$. Furthermore, there are two demotivating paths, consisting of $2h$ vertices, reachable from the last vertex of $P_n^{c_n}(t_n)$. Lastly, the following claim is straightforward to verify.

Lemma 14 *The number of clause vertices that are open after normal play is*

$$d \times (\textit{number of clauses of } \psi \textit{ statisfied by the assignment } (t_1, \ldots, t_n))$$

We summarize our observations as follows:

Lemma 15 *If $(t_1, \ldots, t_n)$ satisfies ψ then the number of open vertices is*

$$2h + nmd + md + 1;$$

otherwise, the number of open vertices is at most

$$2h + nmd + (m-1)d + 1.$$

In the last round, Paul has $L = 2h + nmd + (m-1)d + 1$ questions. The above discussion proves the following result.

Lemma 16 *With normal play, Carole wins G_ψ if and only if the ψ is true.*

Remark 17 *Paul has $2h+nmd+(m-1)d+1$ questions in the last round. If ψ is satisfiable and Paul and Carole play normally then the number of open vertices in the last round is $2h + nmd + md + 1$. Thus Paul has a deficit of $d = 15n$ questions. In Section 5.4 we show that Paul cannot reduce this deficit to 0 by playing non-normally although he maybe able to reduce it somewhat.*

5.3 Non-normal Play for Carole

If ψ is false then Carole has no motivation to play normally as she knows that she will lose. Can she try some non-normal strategy to beat Paul? In this section, we prove that this is not possible. Let us suppose that ψ is false. Note that the initiative is always with Paul as he poses the questions. Carole only has the discretion of answering. Hence, the only non-normal move she can make is when Paul is trying to set the value of a universally quantified variable x_{i+1} in round $2i+1$ where i is odd. Suppose Paul wants to set $x_{i+1} = 1$.

He inquires about the value selection vertex of the current strand and the last vertex on the path $P_{i+1}^{c_{i+1}}(0)$, hoping that Carole will "waste" his second question by answering the first question in the direction of $P_{i+1}^{c_{i+1}}(1)$. Instead, Carole selects the start vertex of $P_{i+1}^{c_{i+1}}(0)$. However, she also answers the last vertex of $P_{i+1}^{c_{i+1}}(0)$ thereby selecting a strand $S_{i+2}^{c_{i+2}}$ for the next variable. From this point onwards, Paul is one round *ahead*. He simply ignores the variable assignments for the rest of the game and inquires alternately about the value of x_{i+2} and then the root of the strand $S_{i+3}^{c_{i+3}}$ of x_{i+3} and so on. After exactly $2n-2$ rounds, he is in the same position as he would be after $2n-1$ rounds if Carole had played normally. The only difference is that he has completely lost control of the assignment $(t_1, \ldots, t_n)$ of variables. In the penultimate round, Paul can now inquire about the last vertex of $P_n^{c_n}(t_n)$ thereby forcing Carole to select *one demotivating path*. This one question eliminates an entire demotivating path. Thus, $h+1$ vertices are made unreachable. Hence, the total number of open vertices is $h + nmd + d \times (\text{number of clauses satisfied by } (t_1, \ldots, t_n)) \leq h + nmd + md < L$.

In the last round, he can inquire about all the open vertices. Hence, he wins the game.

Remark 18 *The demotivating paths ensure that Carole cannot afford to let Paul get ahead by even by one round. This ensures that she plays normally or loses.*

We record the result of this section in the following lemma.

Lemma 19 *If ψ is false, Paul wins.* □

5.4 Non-normal Play for Paul

In this section, we show that if ψ is true then Paul cannot win G_ψ even if he plays non-normally. As Paul has the initiative, he can pose questions *anywhere* in the graph. This gives him considerable freedom and we have to carefully argue that he cannot use this freedom to win the game when ψ is satisfiable. As stated earlier in Remark 17 he has a deficit of d questions if ψ is satisfiable. We show that he cannot bridge this deficit by playing non-normally.

In particular, in round $2i-1$, if i is even, he has two questions. The two questions are used to dictate the value, t_i, of the universally quantified variable x_i. However, he knows that either value, $t_i = 0$ or $t_i = 1$, will lead him to lose the game. Hence, he can ask one question about $v_i^{c_i}$ and the second question somewhere else in the graph. This way he "gives up" his privilege of selecting the value t_i and lets Carole choose the value for him. He can hope to use the second question in a meaningful way. Indeed, asking this question, for example, on the path $P_1^1(t_1)$ is more meaningful. Playing this way may reduce the deficit of questions that he faces in the last round. However, the game is so robust that such moves do not allow Paul to win the game.

When Paul plays non-normally, we argue that Carole can afford to be *generous* in her play at times. However, Carole will never answer more than two questions generously in any round.

We say that $P_i^c(0)$ (resp. $(P_i^c(1))$ is *clean* if $\overline{w}_i^c(md)$ (resp. $w_i^c(md)$) has not been inquired about. For $i < n$, S_i^c is called clean if no vertex in $\{v_i^c, \overline{w}_i^c(md), w_i^c(md)\}$ is not inquired about. S_n^c is considered clean if there are no answered questions on S_n^c and none of the vertices on the four demotivating paths reachable from the root of S_n^c have been inquired about. We say these subgraphs are *dirty*, if they are not clean.

Let us first observe the following simple fact:

Fact 20 *In any round, for all $i = 1, \ldots, n$, there exist c and c' such that both the strands, S_i^c and $S_n^{c'}$, are clean.*

Proof. This is a simple consequence of the pigeonhole principle. Paul asks at most $3n$ questions in the first $2n - 1$. Carole answers at most $4n - 2$ questions generously in the entire game. A question can make at most one strand dirty. Since there are $r = 8n$ strands for each i, thus there are at least $n + 2 \geq 2$ clean strands. $\square$

By a *default* answer to a question we mean the following choices made by Carole when a question v is posed by Paul:

1. If $v = w_i^c(j)$ (resp. $\overline{w}_i^c(j)$) and $j < md$ then Carole points it to $w_i^c(j+1)$ (resp. $\overline{w}_i^c(j+1)$).
2. If v is on a demotivating path or v is a clause vertex then Carole answers it arbitrarily.

Note that the default choices are not defined for the value selection vertices and the output vertices of a strand.

After round $2i - 1$ (equivalently before round $2i$), we say that Paul is *behind* if there exists $c_1, \ldots, c_i$ and $t_1, \ldots, t_{i-1}$ such that:

1. All the vertices on paths $P_1^{c_1}(t_1), \ldots, P_{i-1}^{c_{i-1}}(t_{i-1})$ are reachable.
2. The value selection vertices, $v_1^{c_1}, \ldots, v_{1-1}^{c_{i-1}}$ are answered.
3. $S_i^{c_i}$ is clean and all the vertices of $S_i^{c_i}$ are reachable.

After round $2i$ (equivalently before round $2i + 1$), we say that Paul is *behind* if there exists $c_1, \ldots, c_i$ and $t_1, \ldots, t_i$ such that:

1. All the vertices on paths $P_1^{c_1}(t_1), \ldots, P_i^{c_i}(t_i)$ are reachable.

2. The value selection vertices $v_1^{c_1}, \ldots, v_i^{c_i}$ are answered.

3. $P_i^{c_i}(t_i)$ is clean.

Lemma 21 *If Paul is behind at any point in the game then Carole wins.*

Proof. We first show that if Paul is behind before round $2i-1$ then Carole can make sure that he remains behind after round $2i-1$ for $i \leq n$. Consider the $(2i-1)$-st round. If i is odd Paul has two questions, and if i is even then Paul has only one question. In any case Carole answers Paul's questions using the default strategy, if required. She also answers the last vertex of $P_i^{c_{i-1}}(t_{i-1})$ (possibly generously) to the root of the strand $S_i^{c_i}$ which is clean. Note that such a strand always exists by Fact 20. Thus Paul remains behind after round $2i-1$.

Similarly, we argue that if Paul is behind before round $2i$ then Carole can make sure that he remains behind after round $2i$ for $i < n$. In $2i$-th round, Paul has only one question. If required, she answers Paul's questions using default strategy. If Paul inquires about any vertex in $S_i^{c_i}$ then she answers $v_i^{c_i}$ to the root of $P_i^{c_i}(0)$ or $P_i^{c_i}(1)$ according to where he has not placed his questions (possibly generously). Once again it is easy to see that Paul remains behind after round $2i$.

Thus we may assume that Paul is behind before the last round. Hence, all the vertices on $P_1(t_1), \ldots, P_{n-1}(t_{n-1})$, $S_n^{c_n}$ and the four demotivating paths reachable from the root of $S_n^{c_n}$ are reachable. Furthermore, $S_n^{c_n}$ is clean. Thus a total of $4h + nmd + md$ vertices are reachable in the last round, out of which:

$$\text{the number of open vertices } \geq 4h + nmd + md - 7n > L,$$

and Paul loses. □

Carole can maintain the following invariant before the start of $(2i-1)$-st round and $1 < i \leq n$ or make sure that Paul falls behind.

Invariant: There exists $c_1, \ldots, c_i$ and $t_1, \ldots, t_{i-1}$ such that:

1. $v_j^{c_j}$ point to the root $P_j^{c_j}(t_j)$ for all $j = 1, \ldots, i-1$.

2. if $t_j = 0$ (resp. $t_j = 1$) then $\overline{w}_j^{c_j}(md)$ (resp. $w_j^{c_j}(md)$) points to $v_{j+1}^{c_j+1}$ for all $j = 1, \ldots, i-1$.

3. if $t_{i-1} = 0$ (resp. $t_{i-1} = 1$) then $P_{i-1}^{c_i-1}(0)$ (resp. $P_{i-1}^{c_i-1}(1)$) is clean.

4. For odd values of j and $j \leq i-1$, t_j's are chosen by Carole after knowing the value of $t_1, \ldots, t_{j-1}$.

Clearly, the invariant holds for $i = 1$ with $c_1 = 1$ (i.e., at the beginning of the first round) since no t_i's are chosen and no vertices of S_1^0 are answered. Let us assume that the invariant has been maintained before the beginning of the $(2i-1)$-st round and Paul has not fallen behind. Let us see how this invariant can be maintained after the $(2i-1)$-st round.

Let i be odd. In this case Paul has one question in the round $2i-1$ and one question in the $2i$-th round.

Case 1 Paul does not inquire about any vertex in $\{v_i^{c_i}, \overline{w}_i^{c_i}, w_i^{c_i}(md)\}$. In this case she answers the questions posed by Paul using the default strategy and Paul falls behind.

Case 2 Paul inquires about $v_i^{c_i}$ in round $2i-1$. In this case, Carole chooses t_i and points $v_i^{c_i}$ to $P_i^{c_i}(t_i)$ and waits for Paul to play the second round. If Paul inquires about the last vertex of $P_i^{c_i}(t_i)$ she points it to the root of a strand $S_{i+1}^{c_i+1}$ that is clean. This is possible by Fact 20. If Paul does not inquire about $w_i^{c_i}(md)$ then she uses the default strategy and Paul falls behind.

Case 3 Paul inquires about $w_i^{c_i}(md)$ (resp. $\overline{w}_i^{c_i}(md)$) in round $2i-1$. In this case she points $w_i^{c_i}(md)$ (resp. $\overline{w}_i^{c_i}(md)$) to the root of a clean strand S_{i+1}^{c+1}. If Paul does not inquire about $\overline{w}_i^{c_i}$ (resp. $w_i^{c_i}(md)$) then she can make sure that he falls behind by pointing $v_i^{c_i}$ to $\overline{w}_i^{c_i}(md)$ (resp. $w_i^{c_i}(md)$). If he does inquire about $\overline{w}_i^{c_i}(md)$ (resp. $w_i^{c_i}(md)$) then she picks a t_i and points $v_i^{c_i}$ to the root of $P_i^{c_i}(t_i)$ generously, and maintains the invariant.

For i even, Paul has two questions in round $2i-1$ and one question in the $2i$-th round.

Case 1 Paul inquires about $v_i^{c_i}$. If Paul has also inquired about $w_i^{c_i}(md)$ (resp. $\overline{w}_i^{c_i}(md)$) then Carole points $v_i^{c_i}$ to the root of $P_i^{c_i}(0)$ (resp. $P_i^{c_i}(1)$). She waits for the next round and expects Paul to inquire about $\overline{w}_i^{c_i}(md)$ (resp. $w_i^{c_i}(md)$). If he does not, she uses the default strategy and he falls behind. On the other hand, if he does inquire about $w_i^{c_i}(md)$ (resp. $\overline{w}_i^{c_i}(md)$), she can answers it to a $v_{i+1}^{c_i+1}$ such that the strand $S_i^{c_i+1}$ is clean and maintain the invariant.

Case 2 Paul inquires about both $w_i^{c_i}(md)$ and $\overline{w}_i^{c_i}(md)$. In this case, she picks two different strands S_{i+1}^{c} and $S_{i+1}^{c'}$ that are clean and points $w_i^{c_i}(md)$ and $\overline{w}_i^{c_i}(md)$ to the root of S_{i+1}^{c} and $S_{i+1}^{c'}$, respectively. In the next round if Paul does not inquire about any vertex in S_{i+1}^{c} then Carole points $v_i^{c_i}$ to the root of $P_i^{c_i}(1)$. Otherwise, she points $v_i^{c_i}$ to the root of $P_i^{c_i}(0)$. It is clear that the invariant is maintained.

Case 3 Paul does not inquire about $v_i^{c_i}$ and inquired about at most one of the vertices from $\{w_i^{c_i}(md), \overline{w}_i^{c_i}(md)\}$. We assume that Paul inquires about $w_i^{c_i}(md)$ the other case being similar. Carole uses the default strategy for the other question that Paul has asked. In the next round, if he does not inquire about $\overline{w}_i^{c_i}(md)$ then she points $v_i^{c_i}$ to the root of $P_i^{c_i}(0)$ and Paul falls behind. If Paul does inquire $\overline{w}_i^{c_i}(md)$ then she points $v_i^{c_i}$ to the root of $P_i^{c_i}(0)$ (thereby picking $t_i = 0$ for Paul arbitrarily) and the invariant is maintained.

Lemma 22 *If ψ is true and Paul plays non-normally then Paul loses.*

Proof. Let us consider round $2n-1$. Let us assume that Paul has not fallen behind (otherwise he would loose due to Lemma 21) and the invariant is maintained. Thus the all vertices on the paths, $P_1^{c_1}(t_1), \ldots, P_{n-1}^{c_{n-1}}(t_{n-1})$, and the strand $S_n^{c_n}$ are reachable. Further more $S_n^{c_n}$ is clean. Paul has one question in the penultimate round. If he does not inquire about $v_n^{c_n}$, he can make the root of at most one of the four demotivating paths unreachable. In that case, the number of open vertices in the last round is at least $3h+nmd+md+2>L$, and he loses.

Thus we only have to analyze the case when he inquires about $v_n^{c_n}$. In this case Carole picks her assignment t_n and points $v_n^{c_n}$ to the root of $P_n^{c_n}(t_n)$. As ψ is true we assume that Carole picks t_j's in such a way that $\phi|_{(t_1,\ldots,t_n)}$ is true.

Let us count the total number of open vertices in the game. There are n paths, $P_i^{c_i}(t_i)$ for $i = 1, \ldots, n$, that are reachable. Furthermore, there are at least two demotivating paths, consisting of $2h$ vertices, reachable from the last vertex of $P_n^{c_n}(t_n)$. There can be at most $7n$ vertices which are answered on these paths. A question placed on a path can eliminate at most 2 open vertices, one is the vertex itself and other may be the clause vertex that it leads to. On the other hand, a question placed on the clause vertex can only eliminate one open vertex; that is, the clause vertex itself. All other clause vertices are reachable since $\phi|_{(t_1,\ldots,t_n)}$ is true.

Thus, the number of open vertices is at least $2h+nmd+md-14n > L$ and Paul loses.□

The above analysis shows that Paul does bridge some gap by playing non-normally. However, it is not enough to win the game.

It is not hard to see that G_ψ can be computed from ψ in **LOGSPACE** using an algorithm similar to Algorithm 3.1. Indeed, with some minor changes and addition two counters one catering for r copies of the strands and the other catering for several copies of the clauses allows us to come up with a **LOGSPACE**-algorithm for computing D_ψ from ψ. This shows that $\mathcal{C}$ is **PSPACE**-hard.

Algorithm 2 (SolveQAGame) decides $\mathcal{C}$ in polynomial space. It takes as input a game $G = (D, (q_1, \ldots, q_r))$ and a set Z of internal vertices of D. Note that we ignore specifying the root s of the graph D in the algorithm. A variable *turn* indicates if the next move is to be made by Carole or Paul. In case, it is Carole's move, the questions posed by Paul are given by the set Z. The algorithm return a value *win* from the set {Paul-win, Carole-win}. This establishes Theorem 13.

Acknowledgment

We would like to thank the anonymous referee whose comments helped greatly in improving the exposition of this paper.

```
**ReduceGame**:
**Input**: $(D = (V, E), Q, f)$
**Output**: $(G = (D, \mathbf{q})$
**begin**
    $E' = E \setminus \{(u, v) | u \in Q, v \in N^+(u)$ and $v \neq f(u)\}$;
    $D' = (V, E')$;
    $G' = (D', (q_2, \ldots, q_r))$;
    **return** $G'$;
**end**
**SolveQAGame**:
**Input**: $(G = (D = (V, E), (q_1, \ldots, q_r)), turn, Z)$
**Output**: $(win)$
**begin**
    **if** $turn = Paul$ **then**
        **if** *the number of open vertices in* $G \leq q_1$ **then**
            **return** Paul-win;
        **else**
            **if** $r > 1$ **then**
                **foreach** *set* $Z$ *of* $q_1$ *internal vertices of* $D$ **do**
                    **if** ***SolveQAGame***$(G, Carole, Z)$=*Paul-win* **then**
                        **return** Paul-win;
                    **end**
                    **return** Carole-win;
                **end**
            **else**
                **return** Carole-win;
            **end**
        **end**
    **end**
    **if** $turn = Carole$ **then**
        **foreach** $f : Z \rightarrow V$ **do**
            $G' :=$ **ReduceGame**$(D, Z, f)$;
            **if** ***SolveQAGame***$(G', Paul, \emptyset)$=*Carole-win* **then**
                **return** Carole-win;
            **else**
                **return** Paul-win;
            **end**
        **end**
    **end**
**end**
```

Algorithm 2: Polynomial space algorithm to decide the winner of a Q/A game.

References

[1] S. Abbasi, *Do answers help in posing questions?*, Tech. Report 98-35, DIMACS, 1998.

[2] S. Abbasi, *Impact of parallelism on the efficiency of binary tree search*, Ars Combinatoria (to appear).

[3] B. Bollobas, *Modern Graph Theory*, Springer-Verlag, NewYork, 1998.

[4] S. A. Cook, *The complexity of theorem-proving procedures*, STOC '71: Proceedings of the third annual ACM symposium on Theory of computing (New York, NY, USA), ACM Press, 1971, pp. 151–158.

[5] I. Dumitriu and J. Spencer, *The liar game over an arbitrary channel*, Combinatorica **25** (2005), no. 5, 537–559.

[6] I. Dumitriu and J. Spencer, *The two-batch liar game over an arbitrary channel*, SIAM J. Discret. Math. **19** (2005), no. 4, 1056–1064.

[7] S. Even and R. E. Tarjan, *A combinatorial problem which is complete in polynomial space*, J. ACM **23** (1976), no. 4, 710–719.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractibility: A Guide to the Theory of NP-Completeness*, W.H. Freeman, 1979.

[9] C. Löding and P. Rohde, *Solving the sabotage game is pspace-hard*, MFCS, 2003, pp. 531–540.

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

[11] A. Pelc, *Ulam's problem on searching with a lie*, J. Comb. Theory Ser. A. **44** (1987), 129–140.

[12] A. Pelc, *Searching games with errors—fifty years of coping with liars*, Theor. Comput. Sci. **270** (2002), no. 1-2, 71–109.

[13] J. Spencer, *Ulam's searching game with fixed number of lies*, Theoretical Computer Science **95** (1992).

[14] L. J. Stockmeyer and A. R. Meyer, *Word problems requiring exponential time (preliminary report)*, STOC '73: Proceedings of the fifth annual ACM symposium on Theory of computing (New York, NY, USA), ACM Press, 1973, pp. 1–9.

[15] R. Trakhtenbrot, *A survey of russian approaches to perebor (brute-force search) algorithms*, Annals of the History of Computing **6** (1984), no. 4, 384–400.

[16] S. Ulam, *Adventures of a Mathematician*, Scribner, New York, 1977.