Pinpointing Computation with Modular Queries in the Boolean Hierarchy

Manindra Agrawal * Richard Beigel † Thomas Thierauf ‡

Abstract

A *modular query* consists of asking how many (modulo m) of k strings belong to a fixed NP language. Modular queries provide a form of restricted access to an NP oracle. For each k and m, we consider the class of languages accepted by NP machines that ask a single modular query. Han and Thierauf [HT95] showed that these classes coincide with levels of the Boolean hierarchy when m is even or $k \leq 2m$, and they determined the exact levels. Until now, the remaining case — odd m and large k — looked quite difficult. We pinpoint the level in the Boolean hierarchy for the remaining case; thus, these classes coincide with levels of the Boolean hierarchy for every k and m.

In addition we characterize the classes obtained by using an NP(l) acceptor in place of an NP acceptor (NP(l) is the lth level of the Boolean hierarchy). As before, these all coincide with levels in the Boolean hierarchy.

*Abteilung Theoretische Informatik, Universität Ulm, Oberer Eselsberg, 89069 Ulm, Germany. Email: manindra@informatik.uni-ulm.de. On leave from SPIC Science Foundation, Madras, India. Research supported by the Alexander von Humboldt Fellowship.

†Yale University, Dept. of Computer Science, P.O. Box 208285, New Haven, CT 06520-8285, USA. Email: beigel-richard@cs.yale.edu. Supported by grants CCR-8952528 and CCR-9415410 from the United States National Science Foundation.

‡Abteilung Theoretische Informatik, Universität Ulm, Oberer Eselsberg, 89069 Ulm, Germany. Email: thierauf@informatik.uni-ulm.de.

1 Introduction

A set L is *(polynomial-time) truth-table reducible to* a set A [LLS75], if there exist two polynomial-time bounded Turing machines, the *generator* and the *evaluator*. On a given input string x, the generator first generates a list of strings which are then asked of oracle A. Then the evaluator, getting x and the answers of A to the queries as input, decides the membership of x in L. A truth-table reduction is called *bounded* if the number of queries produced by the generator is bounded by a constant for any x.

In this paper, we consider a more restrictive version of a truth-table reduction. Namely, instead of giving the *full information* about the queries to the evaluator, that is, the characteristic sequence with respect to oracle A, the evaluator gets only some partial information about it. The point is that by comparing various kinds of partial information that can be given to an evaluator, one can study the kind of information an evaluator actually needs to solve a certain problem. This setting has been studied in many papers [Bei91, HT95, KT94, W90, Wec85], and some surprising results have been obtained. To describe this more formally, we use a notation introduced by Köbler and Thierauf [KT94].

Definition [KT94] Let $\mathcal{C}$ be a class of languages and let $\mathcal{F}$ be a class of functions from Σ^* to Σ^*. A set L is in the class $\mathcal{C}//\mathcal{F}$ if and only if there are a set $A \in \mathcal{C}$ and a function $f \in \mathcal{F}$ such that for all $x \in \Sigma^*$, it holds that $x \in L \iff (x, f(x)) \in A$.

We consider the following function classes. Let A be a set, $k \geq 0$, and $m \geq 2$.

$$\begin{aligned}
f \in \chi^{A[k]} &\iff \exists g \in \mathrm{FP}\ \forall x: \quad g(x) = (x_1, \ldots, x_k) \text{ and } f(x) = A(x_1) \cdots A(x_k), \\
f \in \#^{A[k]} &\iff \exists g \in \mathrm{FP}\ \forall x: \quad g(x) = (x_1, \ldots, x_k) \text{ and } f(x) = \sum_{i=1}^{k} A(x_i), \\
f \in \mathrm{Mod}_m^{A[k]} &\iff \exists h \in \#^{A[k]}\ \forall x: \quad f(x) = h(x) \bmod m,
\end{aligned}$$

where $A(\cdot)$ denotes the characteristic function of A. For $m = 2$, we also write $\oplus^{A[k]}$ instead of $\mathrm{Mod}_m^{A[k]}$. For a class $\mathcal{C}$ of sets, $\chi^{\mathcal{C}[k]}$ denotes $\bigcup_{A \in \mathcal{C}} \chi^{A[k]}$, and analogously for the other two classes.

In other words, $\chi^{A[k]}$ gives the sequence of answers to the queries to A produced by some generator g, $\#^{A[k]}$ counts the number of queries that are in A, and $\mathrm{Mod}_m^{A[k]}$ gives the later number modulo m. As an example, we have $\mathrm{P}//\chi^{\mathrm{NP}[k]} = \mathrm{P}_{tt}^{\mathrm{NP}[k]}$.

One of the motivations to consider such restricted truth-table reductions is a somewhat surprising result that follows from a paper by Wagner and Wechsung [Wec85], see also in [Bei91].

Theorem 1.1. [Wec85] For all $k \geq 0$,

$$\mathrm{P}//\chi^{\mathrm{NP}[k]} \;=\; \mathrm{P}//\#^{\mathrm{NP}[k]} \;=\; \mathrm{P}//\oplus^{\mathrm{NP}[k]}.$$

In other words, one can drastically reduce the information a polynomial-time evaluator gets when asking an NP oracle, namely from full information to a single bit information: the parity of the number of queries that are in the oracle, without changing the accepted class of sets, $\mathrm{P}_{tt}^{\mathrm{NP}[k]}$.

It follows from Theorem 1.1 that instead of $\oplus^{\mathrm{NP}[k]}$, one can use $\mathrm{Mod}_m^{\mathrm{NP}[k]}$, for any *even* m, and still get the same class, $\mathrm{P}//\oplus^{\mathrm{NP}[k]}$. This is, however, not clear when m is *odd*. Han and Thierauf [HT95] showed that in this case the evaluator gets in fact less information (unless the Boolean hierarchy collapses).

Theorem 1.2. **[HT95]** For all $k \geq 0$ and $m > 2$ odd,

$$\mathrm{P}//\mathrm{Mod}_m^{\mathrm{NP}[k]} \;=\; \mathrm{P}//\oplus^{\mathrm{NP}[k-\lfloor k/m \rfloor]}.$$

In other words, a parity function can ask $\lfloor k/m \rfloor$ less queries to an NP oracle than a modulo m function, for odd m, and still give the same amount of information to a P evaluator.

Extending the evaluator to a *nondeterministic* machine, we get a nondeterministic version of the truth-table reduction. Köbler and Thierauf [KT94] showed that the counterpart of the first equality of Theorem 1.1 holds, and furthermore, that the resulting class coincides with the $(2k+1)$-th level of the Boolean hierarchy.

Theorem 1.3. **[KT94]** For all $k \geq 0$,

$$\mathrm{NP}//\chi^{\mathrm{NP}[k]} \;=\; \mathrm{NP}//\#^{\mathrm{NP}[k]} \;=\; \mathrm{NP}(2k+1).$$

Note that $\mathrm{NP}(k) \;\subseteq\; \mathrm{P}//\chi^{\mathrm{NP}[k]} \;\subseteq\; \mathrm{NP}(k+1)$ for all $k \geq 1$ [KSW87] (see also [Bei91]). Therefore, switching from a P to an NP evaluator roughly doubles the level of the Boolean hierarchy where the resulting classes are located.

When we have a parity function given to an NP evaluator, Han and Thierauf[HT95] showed that the resulting classes are located much lower in the Boolean hierarchy than with full information.

Theorem 1.4. **[HT95]** For all $k \geq 0$,

$$\mathrm{NP}//\oplus^{\mathrm{NP}[2k+1]} \;=\; \mathrm{NP}//\oplus^{\mathrm{NP}[2k+2]} \;=\; \mathrm{NP}(2k+3).$$

For general modulo functions, again, when m is even, $\mathrm{Mod}_m^{\mathrm{NP}[k]}$ gives the same information to an NP evaluator as $\oplus^{\mathrm{NP}[k]}$. However, when m is odd, only lower and upper bound are known for the resulting classes, the precise location of $\mathrm{NP}//\mathrm{Mod}_m^{\mathrm{NP}[k]}$, when m is odd, remained open.

Theorem 1.5. **[HT95]** For all $k \geq 2m-2$,

(i) $\mathrm{NP}//\mathrm{Mod}_m^{\mathrm{NP}[k]} \;=\; \mathrm{NP}//\oplus^{\mathrm{NP}[k]}$, for m even,

(ii) $\mathrm{NP}//\oplus^{\mathrm{NP}[k-\lfloor k/m \rfloor]} \;\subseteq\; \mathrm{NP}//\mathrm{Mod}_m^{\mathrm{NP}[k]} \;\subseteq\; \mathrm{NP}//\oplus^{\mathrm{NP}[k]}$, for m odd.

In this paper, we solve the open problem. Namely, we show that all classes $\mathrm{NP}//\mathrm{Mod}_m^{\mathrm{NP}[k]}$ in fact coincide with some level of the Boolean hierarchy, and we determine the level. Based on the mind-change technique developed by Wagner and Wechsung [Wec85], we associate with each class $\mathrm{NP}//\mathrm{Mod}_m^{\mathrm{NP}[k]}$ a certain game. The game consists of a table where one can make certain moves according to rules, we will specify below. Some of the moves increase the counter for the game. The maximum score the counter can reach by any playing strategy will be the level of the Boolean hierarchy the class $\mathrm{NP}//\mathrm{Mod}_m^{\mathrm{NP}[k]}$ coincides with.

Therefore, we will develop a playing strategy and then prove that this strategy is optimal. Since we also get again the results mentioned above (with NP evaluators), we have now a uniform way of proving results along these lines. Furthermore, our technique extends to more general classes: the evaluator can be in higher levels of the Boolean hierarchy. That is, we can handle classes $\mathrm{NP}(j)//\mathrm{Mod}_m^{\mathrm{NP}[k]}$, for $j \geq 1$.

Such classes look very technical. However, we think that our methods of locating such, somehow involved classes, in the, much simpler defined, Boolean Hierarchy are interesting enough in its own and might find further applications in other settings.

2 Preliminaries

We follow standard definitions and notations in computational complexity theory (see, e.g., [HU79] or [BDG88]). Throughout this paper, we use the alphabet $\Sigma = \{0, 1\}$.

P (NP) denote the classes of languages that can be recognized by a polynomial-time deterministic (nondeterministic) Turing machine. FP is the class of polynomial-time computable total functions.

The Boolean hierarchy is the closure of NP under Boolean operations, and is usually defined in levels by allowing successively more Boolean operations. This can be done for example by symmetric differences of NP sets [CGH$^+$88]: a set L is in NP(k) ($k \geq 1$), the *k-th level of the Boolean hierarchy*, if there exist $A_1, \ldots, A_k \in$ NP such that $L = A_1 \triangle \cdots \triangle A_k$. A set L is in coNP(k), if $\overline{L} \in$ NP(k). The *Boolean hierarchy*, BH, is the union of all the levels, BH $= \bigcup_{k \geq 1}$ NP(k).

3 Providing the Game

Wagner and Wechsung [Wec85] have shown that any Boolean expression over NP sets coincides with some level (or its complement) of the Boolean hierarchy.

Let α be a Boolean function with k variables, that is $\alpha : (x_1, \ldots, x_k) \in \{0, 1\}^k \mapsto \{0, 1\}$. This corresponds naturally to the above setting: for an expression using k NP sets $L_1, \ldots, L_k$, set variable $x_i = 1$ if and only if input $x \in L_i$, for $i = 1, \ldots, k$. Then, x is in the set described by α and $L_1, \ldots, L_k$ if $\alpha(x_1, \ldots, x_k) = 1$. Let NP($\alpha$) be the class of sets that can be expressed that way. We have that for any α there is some m such that NP(α) coincides with either NP(m) or coNP(m).

Let $a = (a_1, \ldots, a_m)$ be an increasing chain in $\{0, 1\}^k$ with respect to the bitwise order $\prec$. That is, $a_i \in \{0, 1\}^k$, for $i = 1, \ldots, k$, and $a_1 \preceq \ldots \preceq a_m$. The number of mind-changes of α in a is the number of positions i such that $\alpha(a_i) \neq \alpha(a_{i+1})$. By $m(\alpha)$ we denote the maximum number of mind-changes of α in any increasing chain in $\{0, 1\}^k$. The level of the Boolean hierarchy NP(α) coincides with is determined by $m(\alpha)$.

Theorem 3.1. [Wec85] For any k-ary Boolean function α, we have

$$\text{NP}(\alpha) \;=\; \begin{cases} \text{NP}(m(\alpha)), & \text{if } \alpha(0^k) = 0 \\ \text{coNP}(m(\alpha)), & \text{otherwise.} \end{cases}$$

For any k-ary Boolean function α, we have $0 \leq m(\alpha) \leq k$. As a trivial example, we can apply Theorem 3.1 to classes NP(k) themselves. According to the definition, function α associated with NP(k) is the k-ary parity function par$_k$. Clearly, $m(\text{par}_k) = k$. Since par$_k(0^k) = 0$, we get back NP(k) by Theorem 3.1.

As another example, consider classes NP(l)//$\#^{\text{NP}[k]}$. The Boolean function α associated with NP(l)//$\#^{\text{NP}[k]}$ has $k + l(k + 1)$ variables, namely, k variables $x_1, \ldots, x_k$ for the $\#^{\text{NP}[k]}$ function and l variables $y_{i1}, \ldots, y_{il}$, for each potential value $i \in \{0, \ldots, k\}$ of the $\#^{\text{NP}[k]}$ function. Then the value α is par$_l(y_{i1}, \ldots, y_{il})$, if i of the x_j variables are one. To evaluate α, we don't need to know the exact assignment to its variables: α is composed out of symmetric functions, it is enough to know the number of ones in $(x_1, \ldots, x_k)$ and each tuple $(y_{i1}, \ldots, y_{il})$, for $i = 0, \ldots, k$.

Definition A *state* is a $k + 2$ tuple of integers $(i, c_0, c_1, \ldots, c_k)$ such that $0 \leq i \leq k$, $0 \leq c_j \leq l$ for $0 \leq j \leq k$. We refer to the first component of s (the number i) as the *index* of s, and the number c_j as the *counters* of s. When i is the index of s, counter c_i is the *active* counter of s.

In a state $s = (i, c_0, \ldots, c_{m-1})$, the index i denotes the possible number of ones in $(x_1, \ldots, x_k)$ and the counter c_j denotes the possible number of ones in $(y_{j1}, \ldots, y_{jl})$. The function α can now be thought of as acting on states: $\alpha(s) = c_i (\mathrm{mod}\ 2)$.

An increasing chain of assignments becomes now an increasing sequence of states where we want to maximize the number of mind-changes. We reformulate the problem in terms of a game played on a table.

Definition The *Table* consists of $k+1$ columns and $kl+1$ rows. Entry (C, i) of the Table contains all the states $s = (i, c_0, \ldots, c_k)$ with $\sum_{0 \le j \le k} c_j = C$.

For a state $s = (i, c_0, \ldots, c_k)$, the row and column neighbors of s in the table are defined as follows. The *left and right row neighbors* of s are

$$\begin{array}{rcll} {}^{\leftarrow}s & = & (i-1, c_0, \ldots, c_k), & \text{if } i > 0, \text{ and} \\ s^{\rightarrow} & = & (i+1, c_0, \ldots, c_k), & \text{if } i < k, \end{array}$$

respectively. Similarly, The *upper and lower column neighbors* of s are

$$\begin{array}{rcll} s{\uparrow} & = & (i, c_0, \ldots, c_{i-1}, c_i - 1, c_{i+1}, \ldots, c_k), & \text{if } c_i > 0, \text{ and} \\ s{\downarrow} & = & (i, c_0, \ldots, c_{i-1}, c_i + 1, c_{i+1}, \ldots, c_k), & \text{if } c_i < l, \end{array}$$

respectively.

The game is played by a *Player* on the Table. When the game begins, the Player is in the state $s_{init} = (0, 0, \ldots, 0)$ of the Table, and is allowed to make the following two kinds of moves.

Definition Let $s = (i, c_0, \ldots, c_k)$ be a state. A *row move* from s takes the Player to the state $s^{\rightarrow}$. It is defined only when $i < k$. A *column move* from s takes the Player to the state $s{\downarrow}$. It is defined only when $c_i < l$. The game ends when no more move is possible.

The *mind-change* for any move is defined to be 1 if the active counters of the two neighboring states involved in the move are different modulo two, 0 otherwise.

So, for a column move, the mind-change is always one. And for a row move from the state $s = (i, c_0, \ldots, c_k)$, the mind-change equals $(c_i + c_{i+1}) (\mathrm{mod}\ 2)$.

The aim of the Player is to make moves, starting from the state s_{init}, such that the sum of the mind-changes is maximized.

Definition A *playing strategy* $\mathcal{S}$ for the Player is a sequence of moves having exactly k row moves and such that when the Player plays according to the strategy $\mathcal{S}$, it remains within the Table, i.e., no counter of any state reached during the game exceeds l. For the strategy $\mathcal{S}$, we define the mind-change of the strategy, $mc(\mathcal{S})$, as the sum of the mind-changes of its moves.

The *optimal* playing strategy for the maximizer is a strategy $\mathcal{S}$ such that for all other strategies $\mathcal{S}'$, $mc(\mathcal{S}) \ge mc(\mathcal{S}')$. For a strategy $\mathcal{S}$, we shall use $\mathcal{S}(i)$, with $\mathcal{S}(0) = s_{init}$, to denote the state the Player is in immediately after making the i^{th} row move according to the strategy.

It is clear that the mind-change of an optimal strategy equals $m(\alpha)$. As a column move has always a mind-change of one, it is the row moves that have to be carefully made to arrive at the optimal playing strategy.

Definition A state $s = (i, c_0, \ldots, c_k)$ is *bad* if $c_i = c_{i+1} (\mathrm{mod}\ 2)$. A row move from a bad state is a *bad move*. The state s is *good* if $c_i \neq c_{i+1} (\mathrm{mod}\ 2)$. A row move from a good state is a *good move*.

Coming back to classes $\mathrm{NP}(l)//\#^{\mathrm{NP}[k]}$, the optimal strategy is quite easy to see: we repeat doing l column moves and then a row move until we reach the last column, where we do l column moves. Note that in case that l is odd, all our row moves are good, while they are bad

in case that l is even. Hence, we get $k(l+1)+l$ mind-changes, when l is odd, and $l(k+1)$, if l is even. When l is odd, the number of mind-changes matches the upper bound, therefore the strategy is optimal in this case. When l is even, observe that no strategy can make more than l mind-changes in any column including the row move to the next column. This generalizes the result in [KT94].

Theorem 3.2. For $k, l \geq 1$,

$$\mathrm{NP}(l)//\#^{\mathrm{NP}[k]} = \begin{cases} \mathrm{NP}(l(k+1)+k), & \text{if } l \text{ is odd,} \\ \mathrm{NP}(l(k+1)), & \text{if } l \text{ is even.} \end{cases}$$

The argument here was easy because every counter becomes active exactly once, and therefore, it is obvious that a good strategy makes all possible column moves as soon as a counter becomes active. However, when considering classes $\mathrm{NP}(l)//\mathrm{Mod}_m^{\mathrm{NP}[k]}$, this doesn't hold anymore. To adapt the definitions above, we have now only m counters $c_0, \ldots, c_{m-1}$ in a state, and, for any i, the counter $c_{i \bmod m}$ is the active counter. The Table still has $k+1$ columns, but now $ml+1$ rows. Column- and row moves and mind-changes are defined in the same way, just the indices of the counters have to be taken modulo m now. To keep notation clean, we will mostly write c_i instead of $c_{i \bmod m}$.

Hence, after every m row moves the counter c_j becomes active, for every j, $0 \leq j < m$. Therefore, it is no longer obvious that it is a good strategy to make all the column moves whenever a counter becomes active for the first time. In fact, it will turn out that this is indeed not the optimal strategy.

4 Playing the Game: Two Strategies

An optimal strategy should have the minimum number of bad moves as these moves have no mind-change. Also, note that the very first state of the Player, s_{init}, is a bad state. How does the Player avoid making bad moves? Here the column moves come to the help of the Player. Suppose that s is a bad state. If the Player makes a column move from s then the resulting state $s{\downarrow}$ is a good one since the active counter of this state has a different parity than that of s. Now the row move from $s{\downarrow}$ would be a good one. Therefore, a naive strategy for the Player would be to make a column move from any bad state it reaches and then make a row move. However, this does not always work (in fact, it works for only small values of k) since the number of column moves available is limited ($= ml$). So, when k is large, a more involved strategy is needed. Let

$$K = (l-1) \cdot (m-1) \cdot m.$$

We give below two examples of such strategies, these strategies will be shown to be optimal for m even and odd respectively for $k \geq K + 2m$. The optimality of these strategies are proved in the Section 5. Here, we shall just describe the two strategies and give some intuition as to why are they optimal.

4.1 For m even

The Player plays according to the following strategy $\mathcal{A}$.

For even i and $0 \leq i < m$, make $l-1+l(\text{mod } 2)$ column moves from the state $\mathcal{A}(i)$ ($\mathcal{A}(0) = s_{init}$) followed by a row move. And for odd i and $0 \leq i < m$, make $l-1+(l-1)(\text{mod } 2)$ column moves from the state $\mathcal{A}(i)$ followed by a row move. From the state $\mathcal{A}(m)$ onwards, make row moves until the state $\mathcal{A}(k)$ is reached. Now, if the active counter of the state $\mathcal{A}(k)$ does not equal l, then make a column move from $\mathcal{A}(k)$.

For $0 \leq i < m$, the active counter (c_i) of the state $\overleftarrow{\ }\mathcal{A}(i+1)$ equals 1 modulo two if i is even and 0 modulo two if i is odd. Therefore, for $0 \leq i \leq m-2$, the row move from the state $\overleftarrow{\ }\mathcal{A}(i+1)$ to $\mathcal{A}(i+1)$ is good when i is even, and bad when i is odd. The row move from the state $\overleftarrow{\ }\mathcal{A}(m)$ to $\mathcal{A}(m)$ is good since the active counter of $\mathcal{A}(m)$ is already set to 1 modulo two. Now, any two adjacent counters of the state $\mathcal{A}(m)$ have different value modulo two, and therefore, all the row moves after the state $\mathcal{A}(m)$ made by the Player are good ones. Thus, the Player makes exactly $m/2-1$ bad moves in the whole game. As for the number of column moves, for any two adjacent counters of the state $\mathcal{A}(k)$, exactly one of them equals l (and the other to $l-1$). So, the number of column moves made by the Player equals $ml - m/2$ without counting the (possible) move made from the state $\mathcal{A}(k)$.

Therefore, the mind-change of the strategy (without counting the possible column move made from the state $\mathcal{A}(k)$) is

$$k + ml - m/2 - (m/2 - 1) = k + ml - m + 1.$$

It is shown below (see Rule 5 and the associated claim in the next subsection) that the column move from the state $\mathcal{A}(k)$ is made iff the above number and l have different parity. So,

$$mc(\mathcal{A}) = k + ml - m + 1 + (k + ml - m + 1 + l)(\text{mod } 2) = k + ml - m + 1 + (k + l + 1)(\text{mod } 2).$$

The above strategy allows the Player to arrive at a state $\mathcal{A}(m)$ such that all the states to the right of $\mathcal{A}(m)$ in the Table are good. And so, all the row moves can be made without any loss of mind-change. However, to arrive at this state, the Player has to make $m/2-1$ bad moves, and also make at least $m/2-1$ less column moves than maximum possible. When k is small, the Player can, using the naive strategy described in the previous section, avoid making bad moves; however, for large values of k, bad moves cannot be avoided and then this strategy becomes optimal.

4.2 For m odd

When m is odd, the Player cannot arrive at a state—unlike the case when m is even—such that any two adjacent counters of the state have different value modulo two. So, the Player does the next best thing, i.e., arrive at a state such that exactly one pair of adjacent counters have the same value modulo two. However, from this state if only row moves are made then one state out of every m will be a bad state. So, to minimize the bad moves, the Player makes a column move from such bad states as long as possible. Thus, the following is the strategy, $\mathcal{B}$, for the Player:

For $0 \leq i \leq m-3$, from the state $\mathcal{B}(i)$ make a column move followed by a row move if i is even, otherwise just make a row move. From the state $\mathcal{B}(m-2)$ make a column move and then a row move. From any state $\mathcal{B}(i)$, $m-1 \leq i < k$, make a row move if $\mathcal{B}(i)$ is a good state. On the other hand, if $\mathcal{B}(i)$ is a bad state and the active counter of $\mathcal{B}(i)$ is less than l, then make a column move from $\mathcal{B}(i)$ and then a row move. Finally, from the state $\mathcal{B}(k)$, make a column move if its active counter is less than l.

For the above strategy, the state $\mathcal{B}(m) = (m, 1, 0, 1, 0, \ldots, 1, 0, 1, 1, 0)$ and the number of bad moves made so far are $(m-3)/2$. As the Player proceeds after $\mathcal{B}(m)$, the first bad state that it encounters is $\mathcal{B}(2m-3)$. It then makes a column move and proceeds. So, $\mathcal{B}(2m) = (2m, 1, 0, \ldots, 1, 0, 2, 1, 0)$. Similarly, $\mathcal{B}(3m) = (3m, 1, 0, \ldots, 1, 0, 1, 1, 2, 1, 0)$, $\mathcal{B}(4m) = (4m, 1, 0, \ldots, 1, 0, 2, 1, 2, 1, 0)$ etc. Here, the bad state is occurring after every $m-1$ row moves by the Player. So, after $m(m-1)$ row moves from $\mathcal{B}(m)$ the state of the Player would be $(m^2, 2, 1, \ldots, 2, 1, 2, 2, 1)$, i.e., each counter is one more than that in $\mathcal{B}(m)$. The same pattern of moves will now be repeated. So, after K row moves from $\mathcal{B}(m)$, the state of the Player will be $(m+K, l, l-1, \ldots, l, l-1, l, l, l-1)$. Now, no more column moves are made, except at the end. Also, the Player from now onwards, will encounter a bad state after every m row moves (instead of $m-1$) and it will have to make a bad move then.

Let $d = \lfloor k/m \rfloor$. Then, the number of times that the Player makes a bad move after the state $\mathcal{B}(m+K)$ is $d - (l-1) \cdot (m-1) - 1$ if $k - dm < m-2$, one more otherwise. Also, note that the number of column moves made is exactly $ml - (m-1)/2$ excluding the (possible) column move from the state $\mathcal{B}(k)$. So, the mind-change of the strategy, without counting the possible last column move, is

$$\begin{aligned} & k + ml - (m-1)/2 - (m-3)/2 - (d - (l-1) \cdot (m-1) - 1 + \lfloor (k+2)/m \rfloor - d) \\ = \; & k + 2ml + 4 - 2m - l - \lfloor (k+2)/m \rfloor. \end{aligned}$$

Adding the correction for the last column move (as noted above), we have

$$mc(\mathcal{B}) = k + 2ml + 4 - 2m - l - \lfloor (k+2)/m \rfloor + (k + \lfloor (k+2)/m \rfloor) (\text{mod } 2).$$

5 Proving the Strategies Optimal

In this section, we shall obtain an upper bound on the mind-change of any optimal strategy. We shall concern ourselves with values of $k \geq K + 2m$. For this case, in the previous section, two different strategies are given—one for m even and the other for m odd. We shall show that the upper bound that we derive matches with the mind-changes for these strategies, thus proving them optimal.

Fix an optimal strategy $\mathcal{S}'$. To prove the desired upper bound on $mc(\mathcal{S}')$, we proceed in stages.

Stage 1: Normalizing the Strategy

The column moves in $\mathcal{S}'$ can be arbitrarily distributed between the row moves, and this makes the direct counting of the mind-change of $\mathcal{S}'$ a difficult task. So, we shall rearrange the column moves in $\mathcal{S}'$ according to certain rules to obtain a new strategy such that the column moves in the new strategy have certain 'order' and given the mind-change of the new strategy, the mind-change of $\mathcal{S}'$ can be easily calculated. We call this process, the *normalization* of $\mathcal{S}'$.

Some of the rules that can be used to rearrange the column moves of $\mathcal{S}'$ are evident: from a good state, the Player should not make a column move and then a row move as this renders the row move bad. So such column moves, if existing in $\mathcal{S}'$, should be 'shifted'. Also, any two consecutive column moves in the strategy can be deleted causing a loss of exactly two mind-changes, as the counters in the state of the Player after these two moves have the same value modulo two as the counters in the state before the moves. Using these rules, and a few others, we get a strategy that satisfies the following properties (see Appendix for details).

Lemma 5.1. Let $\mathcal{S}$ be the strategy obtained by normalizing $\mathcal{S}'$. Let b be the number of the pairs of consecutive column moves deleted during the normalization. Suppose that the Player plays according to $\mathcal{S}$. Then,

1. $mc(\mathcal{S}') = mc(\mathcal{S}) + 2b + (mc(\mathcal{S}) + l) (\text{mod } 2)$.
2. Between any two consecutive row moves, the Player makes at most one column move. Also, it makes no column move after the last row move and exactly one column move before the first row move.
3. If a state is good then the Player makes a row move from the state.

Stage 2: Dividing the Column Moves in Two Types

After normalizing $\mathcal{S}'$, we proceed to count the mind-change of the resulting strategy, say $\mathcal{S}$. Let us assume that b is the number of pairs of consecutive column moves deleted from $\mathcal{S}'$ during the normalization. It follows that in $\mathcal{S}$, the maximum number of column moves made is $ml - 2b$. By the above lemma, we have that the Player makes column moves only when its state is bad. Therefore, the number of bad states of the Player during the game are crucial since the difference of this number and the number of column moves made by the Player gives the bad moves in the strategy. For this reason, we associate with each state s of the Player, a number that estimates the number of bad states of the Player during the next m row moves.

For any state $s_0 = (i, c_0, \ldots, c_{m-1})$ of the Player, we consider the $m-1$ states, s_1, s_2, …, s_{m-1} to the right of s_0 in the Table, i.e., $s_j = {s_{j-1}}^{\rightarrow}$ for $1 \leq j < m$. Define $b(s_0)$ to be the number of bad states amongst s_0, …, s_{m-1}. The following lemma lists some properties of $b(s_0)$ (see Appendix for a proof).

Lemma 5.2. Let $s_0 = (i, c_0, \ldots, c_{m-1})$.

1. $b(s_{init}) = m$.
2. If the Player makes a row move from s_0 then $b({s_0}^{\rightarrow}) = b(s_0)$ and $b(s_0)$ is the number of bad states of the Player between, and including, s_0 and $\mathcal{S}(i + m - 1)$.
3. If the Player makes a column move from s_0 then either (1) $b(s_{0\downarrow}) = b(s_0) - 2$ and the row move to the state s_0 (when $i > 0$) is bad, or (2) $b(s_{0\downarrow}) = b(s_0)$ and the row move to the state s_0 (when $i > 0$) is good. Further, $b(s_0)$ is the number of bad states between, and including, $s_{0\downarrow}$ and $\mathcal{S}(i + m - 1)$. Moreover, the Player makes the first kind of moves only if $i < m$.

The above lemma suggests that we can divide the column moves made by the Player into two types.

Reducing column moves. Suppose that the Player, from the state $s_0 = (i, c_0, \ldots, c_{m-1})$, makes a column move and $b(s_{0\downarrow}) = b(s_0) - 2$. This column move reduces the number of bad states the Player encounters during the next m row moves and thus is a very useful move. It is called a *reducing* column move. As $b(s_0) \leq m$ for any state, there cannot be more than $\lfloor m/2 \rfloor$ reducing column moves possible. Also, note that if the column move is reducing, then the row move to the state s_0 must be bad (if it exists, i.e., $i \geq 1$). So, just before making a reducing column move, the Player makes a bad move. The only exception is the first state s_{init} from which a reducing column move is made without making a bad move.

If the aim is to make no bad move, then these column moves, except from s_{init}, should be made. However, for large k—when bad moves cannot be avoided—these moves serve a very useful purpose as illustrated by the two examples in the previous section. In the strategy $\mathcal{A}$ there, $m/2$ reducing column moves are made, so that $b(\mathcal{A}(m)) = 0$, which implies that there are no more bad states encountered during the row moves after $\mathcal{A}(m)$. In the strategy $\mathcal{B}$, $(m-1)/2$ reducing column moves are made, so that $b(\mathcal{B}(m)) = 1$. This leaves some bad states to be taken care of, which is done by the second kind of column moves described below.

Non-reducing column moves. Suppose that the Player, from the state $s_0 = (i, c_0, \ldots, c_{m-1})$, makes a column move and $b(s_{0\downarrow}) = b(s_0)$. This column move does not reduce the number of bad states of the Player. However, it allows the Player to 'postpone' making the bad move by $m-1$ row moves and therefore, still serves a useful purpose. It is called a *non-reducing* column move. There can be as many non-reducing column moves as permitted by the bounds on the counters of the states.

The strategy $\mathcal{B}$ of the previous section uses these moves. As noted above, $b(\mathcal{B}(m)) = 1$, and so, after the state $\mathcal{B}(m)$, whenever the Player encounters a bad state, it uses a non-reducing column move to 'postpone' making the bad move for as long as possible.

Stage 3: Counting the Mind-change of $\mathcal{S}$

Suppose that the Player makes c reducing column moves, while following the strategy $\mathcal{S}$. Since $b(s_{init}) = m$, $b(s) \geq m - 2c$ for any state s of the Player. Also, since the Player makes all the reducing column moves before the first m row moves (Lemma 5.2), $b(\mathcal{S}(m)) = m - 2c$. Since every reducing column move must be preceded by a bad move—except when it is made from state s_{init}—the Player must make $c-1$ bad moves to 'set up' these column moves.

Let $d = \lfloor k/m \rfloor$. We divide the sequence of the states of the Player during the whole game in four segments, S^1, S^2, S^3 and S^4. The first segment, S^1 has states from $\mathcal{S}(0) = s_{init}$ to $\overleftarrow{\ }\mathcal{S}(m)$. The second segment is the smallest one—it has states from $\mathcal{S}(m)$ to $\overleftarrow{\ }\mathcal{S}(m+k-dm)$. The third segment, S^3, has states from $\mathcal{S}(m+k-dm)$ to $\overleftarrow{\ }\mathcal{S}(k-m)$ and the last segment, S^4, has states from $\mathcal{S}(k-m)$ to $\overleftarrow{\ }\mathcal{S}(k)$ (we disregard the state $\mathcal{S}(k)$ as no move is made from it). Since $d \geq 2$, we have $k - m \geq m + k - dm$, i.e., the the second and the last segments do not overlap. We shall count the bad moves in the four segments separately.

First consider the segment S^4. The Player makes exactly m row moves while it is in the states of S^4. It can be shown that,

Lemma 5.3. For $k \geq K + 2m$, the Player makes no column moves while in segment S^4.

Proof. Suppose that the Player makes a column move from the state $\mathcal{S}(k-m+j)$ for some j, $0 \leq j < k$. Then, by the Lemma 5.9.3, the Player makes a non-reducing column move from the state $\mathcal{S}(k-m+j-r(m-1))$ for every r, $1 \leq r \leq \lfloor (k-m+j)/(m-1) \rfloor$. So, there are at least $r+1 \geq \lfloor (k-1)/(m-1) \rfloor \geq (l-1) \cdot m + 2$ non-reducing column moves in the strategy. However, there are only $ml - 2c$ non-reducing column moves available, and—as shown in Section 5—the value of $2c$ is at least $m-1$. This contradicts our assumption. □

Since the values of k that we are interested in satisfy the above bound, there are no column moves by the Player while in S^4. This implies that the Player makes $m - 2c$ bad moves after the state $\mathcal{S}(k-m)$ since $b(\mathcal{S}(k-m)) = m - 2c$.

We can also calculate now the exact number of column moves made by the Player during the entire game. Take any two adjacent good states in S^4. The active counters of these states must be different modulo two. Now consider the original strategy $\mathcal{S}'$. All the counters of the

state $\mathcal{S}'(k)$ must be at least $l-1$ since it is optimal. Also, since the column moves are deleted only in pairs during normalization, the parity of the counters is the same for $\mathcal{S}'(k)$ and $\mathcal{S}(k)$. Therefore, the number of column moves in $\mathcal{S}$ is exactly $ml-2b-c$.

For the first three segments, the counting of the bad moves cannot be done directly. This is because we cannot say beforehand when the Player makes a column move from a bad state and when not. So, we count the number of bad states in these segments and subtract the number of column moves made by the Player from it. This would give us the number of bad moves in the these segments.

First take the segment S^1. The Player makes exactly m row moves while it is in the states of S^1. Also, by Lemma 5.1, it makes all the c reducing column transitions in this segment. Since we have already counted the bad moves made just before reducing column moves, to avoid recounting, we mark the $c-1$ bad states from which the bad moves are made. So, there are exactly $m-c$ unmarked bad states in this segment.

Now take the third segment. All column moves in this, and the second segment must be non-reducing (by Lemma 5.2). If there is a column move from the state $\mathcal{S}(m+k-rm)$ for any r, $1 \le r < d-1$, then, by Lemma 5.2, the number of bad states between $\mathcal{S}(m+k-rm)$ and $\mathcal{S}(m+k-rm+m-1)$ is exactly $m-2c+1$. On the other hand, if there is none, then this number is exactly $m-2c$. Therefore, the bad states in the segment are exactly $(d-2)\cdot(m-2c)+a'$ where a' is the number of column moves made from the states $\mathcal{S}(m+k-rm)$ for $1 \le r < d-1$. The following are the bounds for a' (see Appendix for a proof):

Lemma 5.4. If $2c = m$ then $a' = 0$, otherwise $l - \lceil 2b/m \rceil - m + 2c \le a' \le l-1$.

Finally, we consider the second segment. The Player makes $k-dm < m$ row moves while it is in the states of this segment. Again, we count the bad states in the segment. To minimize such states, the strategy should have as many good states as possible in the segment. The following is the bound on the bad states in the segment (see Appendix for a proof):

Lemma 5.5. S^2 has at least $\max\{0, \min\{m-2c, k+2-dm-2c\}\}$ bad states.

Therefore, the total number of (unmarked) bad states in the first three segments is at least

$$B = m - c + (d-2)\cdot(m-2c) + a' + \max\{0, \min\{m-2c, k+2-dm-2c\}\}.$$

Since the total number of column moves made by the Player is $ml-2b-c$, the number of bad moves made in the first three segments is at least $c-1+\max\{0, B-lm+2b+c\}$. The Player also makes $m-2c$ bad moves in the last segment. So, the total number of bad moves by the Player is at least

$$m-2c+c-1+\max\{0, B-lm+2b+c\} = m-c-1+\max\{0, B-lm+2b+c\}.$$

And therefore,

$$\begin{aligned} mc(\mathcal{S}) &\le k+lm-2b-c-m+c+1-\max\{0, B-lm+2b+c\} \\ &= k+lm-m-2b+1-\max\{0, B-lm+2b+c\}. \end{aligned}$$

Since $d \ge 2$, this expression is maximized if c is chosen as large as possible. Since we have the bound $2c \le m$, we have two cases: m even and odd.

m **is even** : In this case, we can have $2c = m$. Therefore, $a' = 0$ (by Lemma 5.4). So,

$$\begin{aligned} mc(\mathcal{S}) &\leq k + lm - m - 2b + 1 - \max\{0, m - lm + 2b + \max\{0, \min\{0, k + 2 - dm - m\}\}\} \\ &= k + lm - m - 2b + 1 - \max\{0, m - lm + 2b\} \\ &\leq k + lm - m - 2b + 1. \end{aligned}$$

Therefore, for the strategy $\mathcal{S}'$ we have (by Lemma 5.1),

$$\begin{aligned} mc(\mathcal{S}') &\leq k + lm - m - 2b + 1 + 2b + (k + lm - m - 2b + 1 + l) (\text{mod } 2) \\ &= k + lm - m + 1 + (k + 1 + l) (\text{mod } 2) \text{ (since } m \text{ is even).} \end{aligned}$$

The strategy $\mathcal{A}$ described in the previous section achieves this bound, and therefore is optimal.

m **is odd** : In this case, we can only have $2c = m - 1$. So, $l - 1 - \lceil 2b/m \rceil \leq a' \leq l - 1$. Then,

$$\begin{aligned} mc(\mathcal{S}) &\leq k + lm - m - 2b + 1 \\ &\quad - \max\{0, m - lm + d - 2 + 2b + a' + \max\{0, \min\{1, k - dm + 3 - m\}\}\} \\ &= k + lm - m - 2b + 1 \\ &\quad - \max\{0, m - lm + d - 2 + 2b + a' + \lfloor (k+2)/m \rfloor - d\} \text{ (verify!)} \\ &= k + lm - m - 2b + 1 - \max\{0, m - lm - 2 + 2b + a' + \lfloor (k+2)/m \rfloor\}. \end{aligned}$$

Since $k \geq (l-1) \cdot (m-1) \cdot m + 2m$, we have that $\lfloor (k+2)/m \rfloor \geq lm - m + 3 - l$, and therefore,

$$\begin{aligned} mc(\mathcal{S}) &\leq k + lm - m - 2b + 1 - m + lm + 2 - 2b - a' - \lfloor (k+2)/m \rfloor \\ &= k + 2lm - 2m - a' + 3 - 4b - \lfloor (k+2)/m \rfloor. \end{aligned}$$

So, for the strategy $\mathcal{S}'$, we have (by Lemma 5.1),

$$\begin{aligned} mc(\mathcal{S}') &\leq k + 2lm - 2m - a' + 3 - 4b - \lfloor (k+2)/m \rfloor \\ &\quad + 2b + (k - a' + 3 - \lfloor (k+2)/m \rfloor + l) (\text{mod } 2) \\ &\leq k + 2lm + 3 - 2m - l + 1 + \lfloor 2b/m \rfloor - 2b \\ &\quad - \lfloor (k+2)/m \rfloor + (k + a' + 1 + l + \lfloor (k+2)/m \rfloor) (\text{mod } 2). \end{aligned}$$

This expression is maximized when $b = 0$. This implies $a' = l - 1$, and then,

$$mc(\mathcal{S}') \leq k + 2lm + 4 - 2m - l - \lfloor (k+2)/m \rfloor + (k + \lfloor (k+2)/m \rfloor) (\text{mod } 2).$$

The strategy $\mathcal{B}$ described in the previous section achieves this bound, and therefore is optimal.

We summarize the results in the following theorem.

Theorem 5.6. For $k \geq K + 2m$, $\text{NP}(l) // \text{Mod}_m^{\text{NP}[k]} = \text{NP}(t)$, where

$$t = \begin{cases} k + lm - m + 1 + (k + 1 + l) (\text{mod } 2) & \text{if } m \text{ is even,} \\ k + 2lm - 2m - l - \lfloor (k+2)/m \rfloor + 4 + (k + \lfloor (k+2)/m \rfloor) (\text{mod } 2) & \text{otherwise.} \end{cases}$$

Corollary 5.7. For $k \geq 2m$, $\mathrm{NP}//\mathrm{Mod}_m^{\mathrm{NP}[k]} = \mathrm{NP}(t)$, where

$$t = \begin{cases} k+1+k(\mathrm{mod}\ 2) & \text{if } m \text{ is even,} \\ k - \lfloor (k+2)/m \rfloor + 3 + (k + \lfloor (k+2)/m \rfloor)(\mathrm{mod}\ 2) & \text{otherwise.} \end{cases}$$

For values of k less than $K + 2m$, the mind-change of an optimal strategy can be derived in a similar way. When $l = 1$, this was done in [HT95], and for the general case, we have:

Lemma 5.8. For $2m \leq k \leq K + 2m$, and for any optimal strategy $\mathcal{S}'$,

$$mc(\mathcal{S}') \leq k + 1 + ml - m + a + [k + 1 + ml - m + a + l(\mathrm{mod}\ 2)]$$

where a is the largest number satisfying the following conditions.

- $0 \leq a \leq m - 2$.
- $a = m(\mathrm{mod}\ 2)$.
- $a \leq m \cdot (l-1)/\lfloor k/(m-1) \rfloor$.

A strategy $\mathcal{C}$ that achieves the above bound is one in which the Player makes $(m-a)/2$ reducing column moves in the first segment (segment as defined in Section 5), and then a reducing column moves for every $m-1$ row moves after the state $\mathcal{C}(k)$. We omit the calculations.

For $m \leq k \leq 2m$, it can be easily shown that the optimal strategy has a mind-change of $ml + m - 1$ when l is odd, and $ml + 2k - 2m$ when l is even. And for $k < m$, it is derived in Section 3 (in this case, $\mathrm{NP}//\mathrm{Mod}_m^{\mathrm{NP}[k]}$ is the same as $\mathrm{NP}//\#^{\mathrm{NP}[k]}$).

References

[BDG88] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 1988.

[Bei91] R. Beigel. Bounded queries to SAT and the boolean hierarchy. *Theoretical Computer Science*, 84:199–223, 1991.

[CGH+88] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy I: Structural properties. *SIAM Journal on Computing*, 17(6):1232–1252, 1988.

[HT95] T. Han and T. Thierauf. In *Proceedings of the 10th Conference in Structure in Complexity Theory*, pages 206–213, 1995.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[KSW87] J. Köbler, U. Schöning, and K. Wagner. The difference and truth-table hierarchies of NP. *R.A.I.R.O. Informatique théorique et Applications*, 21(4):419–435, 1987.

[KT94] J. Köbler and T. Thierauf. Complexity-restricted advice functions. *SIAM Journal on Computing*, 23(2):261–275, 1994.

[LLS75] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–124, 1975.

[W90] K. Wagner. Bounded query classes. *SIAM J. on Computing* 19(5), pages 833-846, 1990.

[Wec85] G. Wechsung. On the boolean closure of NP. In *Proceedings of the 5th Conference on Fundamentals of Computation Theory*, pages 485–493. Springer-Verlag *Lecture Notes in Computer Science #199,* 1985. (An unpublished precursor of this paper was coauthored by K. Wagner).

Appendix

The Appendix is divided in two subsections.

A. Normalizing an optimal strategy

Here we give rules to normalize an optimal strategy. These rules move/delete certain column moves of the strategy to obtain the normalized strategy.

Let $\mathcal{S}'$ be any optimal strategy and suppose that the Player plays according to it.

Rule 1 (making a column move from the first state): Suppose that there is no column move from the state $\mathcal{S}(0)$. Let there be a column move in $\mathcal{S}'$ after the first j row moves. Let $\mathcal{S}$ be the strategy obtained by deleting the first j row moves of $\mathcal{S}'$ and adding j row moves at the end. Then,

Claim $mc(\mathcal{S}) = mc(\mathcal{S}')$.

Proof. Since the first j row moves of $\mathcal{S}'$ are all bad, $mc(\mathcal{S}) \geq mc(\mathcal{S}')$ and since $\mathcal{S}'$ is optimal, $mc(\mathcal{S}) \leq mc(\mathcal{S}')$. □

Rule 2 (elimination of column moves from good states): Let the Player make a column move from a good state $s = \mathcal{S}(i)$ followed by a row move. If $i \leq k - m$, then let $t = \mathcal{S}(i + m)$, and $\mathcal{S}$ be the strategy obtained from $\mathcal{S}'$ by removing the column move from s and introducing a column move to t. If $i > k - m$, then let $\mathcal{S}$ be the strategy obtained from $\mathcal{S}'$ by removing the column move from s. Then,

Claim $mc(\mathcal{S}) = mc(\mathcal{S}')$.

Proof. The strategy $\mathcal{S}$ gains a mind-change over $\mathcal{S}'$ due to the row move from s, a good state (in $\mathcal{S}'$ this row move is from the state $s{\downarrow}$, a bad state). On the other hand, when $i \leq k - m$, $\mathcal{S}$ loses a mind-change if the row move to the state $t{\uparrow}$ is a bad one. When $i > k - m$, $\mathcal{S}$ loses a mind-change because it has one column less than $\mathcal{S}'$. The rest of the row moves of $\mathcal{S}$ retain their goodness/badness in either case. So, $mc(\mathcal{S}) \geq mc(\mathcal{S}')$ and since $\mathcal{S}'$ is optimal, $mc(\mathcal{S}) \leq mc(\mathcal{S}')$. □

Rule 3 (shifting of column moves from bad states): Let the Player make a column move from a state $s = \mathcal{S}(i)$ with $i \geq m$. Also, suppose that either the state $t = {}^{\leftarrow}\mathcal{S}(i - m + 1)$ or the state ${}^{\leftarrow}\mathcal{S}(i)$ is a bad state. Let $\mathcal{S}$ be the strategy obtained from $\mathcal{S}'$ by removing the column move from s and introducing a column move from t. Then,

Claim $mc(\mathcal{S}) = mc(\mathcal{S}')$.

Proof. If the state t is bad, then the strategy $\mathcal{S}$ gains a mind-change over $\mathcal{S}'$ due to the row move from $t{\downarrow}$, a good state (in $\mathcal{S}'$ this row move is from the state t). It loses a mind-change if the row move to the state $s{\downarrow}$ is a bad one. On the other hand, if the state ${}^{\leftarrow}\mathcal{S}(i)$ is bad, then the row move to the state $s{\downarrow}$ is a good one in the strategy $\mathcal{S}$ thus adding an extra mind-change to the strategy. However, the strategy may lose a mind-change if the state t was good since the row move from $t{\downarrow}$ becomes bad in that case. The rest of the row moves of $\mathcal{S}$ retain their goodness/badness. Therefore, $mc(\mathcal{S}) \geq mc(\mathcal{S}')$, and since $\mathcal{S}'$ is optimal, $mc(\mathcal{S}) \leq mc(\mathcal{S}')$. □

Rule 4 (deletion of two consecutive column moves): For this rule, we do not need the assumption that $\mathcal{S}'$ is an optimal strategy. Suppose that the Player makes two consecutive column moves from a state s. Let $\mathcal{S}$ be the strategy obtained from $\mathcal{S}'$ by deleting these two moves. Then,

Claim $mc(\mathcal{S}) = mc(\mathcal{S}') - 2$.

Proof. All the counters of the states s and $s_{\downarrow\downarrow}$ have identical value modulo two. Therefore, the row moves of $\mathcal{S}$ will retain their goodness/badness. So, the mind-change of $\mathcal{S}$ is exactly two less than that of $\mathcal{S}'$ as it has two less column moves. □

Rule 5 (deleting a column move from $\mathcal{S}'(k)$): For this rule also, we do not make the assumption that $\mathcal{S}'$ is an optimal strategy. Rather, we make the assumption that the Player makes a column move from the state $\mathcal{S}'(k)$ iff the active counter of $\mathcal{S}'(k)$ equals $l-1$ modulo two. Let $\mathcal{S}$ be the strategy obtained by deleting a single column move (if there is any) from the moves made by the Player from the state $\mathcal{S}'(k)$. The following claim describes the relationship between mind-change of $\mathcal{S}$ and $\mathcal{S}'$. It is also used to add the correction for the (possible) last column move in the strategies $\mathcal{A}$ and $\mathcal{B}$ in the Section 4.

Claim $mc(\mathcal{S}') = mc(\mathcal{S}) + [(l + mc(\mathcal{S}))(\text{mod } 2)]$.

Proof. It is easy to verify that when the Player is in the state s, then the sum of the mind-changes of all the moves up to s has the same value modulo two as the active counter of s. Since the Player makes a column move from $\mathcal{S}'(k)$ iff the active counter of the state equals $l-1$ modulo two, the claim follows. □

Now we define the normalized form of a strategy.

Definition For a strategy $\mathcal{S}'$, the *normalization* of $\mathcal{S}'$ is a strategy, $\mathcal{S}$, that is obtained by first shifting/deleting the moves in $\mathcal{S}'$ as specified in the first three Rules above until no more movement/deletion is possible. Then, deleting the column moves according to the Rules 4 and 5. The strategy $\mathcal{S}$ is called a *normalized* strategy.

The following lemma lists some basic properties of a normalized strategy.

Lemma 5.1 Let $\mathcal{S}$ be the strategy obtained by normalizing $\mathcal{S}'$. Let b be the number of the pairs of consecutive column moves deleted during the normalization. Suppose that the Player plays according to $\mathcal{S}$. Then,

1. $mc(\mathcal{S}') = mc(\mathcal{S}) + 2b + (mc(\mathcal{S}) + l)(\text{mod } 2)$.
2. Between any two consecutive row moves, the Player makes at most one column move. Also, it makes no column move after the last row move and exactly one column move before the first row move.
3. If a state is good then the Player makes a row move from the state.

Proof. The b pairs of consecutive columns moves can only be deleted by the Rule 4. Since $\mathcal{S}'$ is an optimal strategy, the active counter of the final state of the Player, following $\mathcal{S}'$, must equal l. So, even after the deletion of column moves by Rule 4, the active counter of the last state of the Player, following the new strategy, say $\mathcal{S}''$, will have the same value modulo two as

l. Therefore, a column move is deleted by Rule 5 iff the active counter of $\mathcal{S}''(k)$ equals $l-1$ modulo two. By the Claim for Rule 5, the first part follows.

The second and third parts follow directly from the definition of a normalized strategy.

□

The following lemma lists some technical properties of a normalized strategy that will be used in the proofs of the lemmas in the next subsection.

Lemma 5.9. Let $\mathcal{S}'$ be any optimal strategy and $\mathcal{S}$ be the strategy obtained by normalizing $\mathcal{S}'$. Suppose that the Player plays according to $\mathcal{S}$. Let $s = \mathcal{S}(i)$ for any $0 \leq i < k$. Then,

1. If $i \geq m$ and the Player makes a column move from s, then the state $\overleftarrow{\ }s$ is a good state.
2. If $i \leq k-m$, the states s and $\mathcal{S}(i+1)$ are bad and good respectively, and the Player does not make a column move from s, then the state $\mathcal{S}(i+rm)$ is a bad state and the Player does not make a column move from $\mathcal{S}(i+rm)$ for every $1 \leq r \leq \lfloor (k-i)/m \rfloor$.
3. If $i \geq m$ and the Player makes a column move from s, then for every r, $1 \leq r \leq \lfloor i/(m-1) \rfloor$, the Player makes a non-reducing column move from the state $\mathcal{S}(i-r(m-1))$.
4. The Player makes a column move from the state $\mathcal{S}(m-2)$.

Proof.

1. This follows from the fact that if $\overleftarrow{\ }s$ is a bad state then the Rule 3 of the normalization would prephone the column move to $\overleftarrow{\ }\mathcal{S}(i-m+1)$ when $i \geq m$.
2. If the property is not true then there must be a smallest r, $1 \leq r \leq \lfloor (k-i)/m \rfloor$ such that $\mathcal{S}(i+rm)$ is a bad state and the Player makes a column move from it or the state $\mathcal{S}(i+rm)$ is a good state. If $\mathcal{S}(i+rm)$ is a good state, then since there is no column move from the bad state $\mathcal{S}(i+rm-m)$, there must have been a column move from the state $\mathcal{S}(i+rm-m+1)$. This is possible, by part 1 of the lemma only if $i+rm-m+1 < m$, which implies that $rm = m$. However, this is not possible since by assumption the state $\mathcal{S}(i+1)$ is good.

   Now suppose that the state $\mathcal{S}(i+rm)$ is a bad state and the Player makes a column move from it. Since r is the smallest such number, we know that the state $\mathcal{S}(i+rm-m)$ cannot be a good state. But then, by the Rule 3 of the normalization, this column move would be shifted to the state $\mathcal{S}(i+rm-m)$.
3. State s must be a bad state since there is a column move from it. Consider the state $\overleftarrow{\ }\mathcal{S}(i-m+1)$. It must be a good state otherwise, by Rule 3, the column move from s would have been prephoned. Let $s = (i, c_0, \ldots, c_{m-1})$ and $\overleftarrow{\ }\mathcal{S}(i-m+1) = (i-m, b_0, \ldots, b_{m-1})$. Since $b_i = c_i$, $c_i = c_{i+1} (\mathrm{mod}\ 2)$ and $b_i \neq b_{i+1} (\mathrm{mod}\ 2)$, $b_{i+1} \neq c_{i+1} (\mathrm{mod}\ 2)$. Therefore, there must be a column move from the state $\mathcal{S}(i-m+1)$ which must be a bad state. And this column move must be non-reducing as the state $\overleftarrow{\ }\mathcal{S}(i-m+1)$ is good. Now the same argument can be repeated for the state $\mathcal{S}(i-m+1)$.
4. Suppose not. Since the Player makes a column move from the state $\mathcal{S}(0)$, the state $\mathcal{S}(m-1)$ is a good state. Now, the fact that there is no column move made from the state $\mathcal{S}(m-2)$ implies that all the states $\mathcal{S}(m-2+rm)$ for $r \geq 1$, must be bad and the Player makes no column move from any of these states (by part 2 of the lemma). Let $\mathcal{S}''$ be the strategy obtained from $\mathcal{S}'$ by applying just the first three Rules above. ($\mathcal{S}$ is the strategy obtained

from $\mathcal{S}''$ by applying the last two Rules). Since $\mathcal{S}''$ is an optimal strategy and (partially) normalized using the first three Rules, there must be an odd number of column moves from the state $\mathcal{S}''(m-2)$. Now, since the Rule 4 deletes column moves in pairs only, there must be a column move from $\mathcal{S}(m-2)$.

□

B. Proofs of some lemmas

In this subsection, we prove some lemmas that are used in the paper.

Lemma 5.2 Let $s_0 = (i, c_0, \ldots, c_{m-1})$.

1. $b(s_{init}) = m$.
2. If the Player makes a row move from s_0 then $b(s_0^{\rightarrow}) = b(s_0)$ and $b(s_0)$ is the number of bad states of the Player between, and including, s_0 and $\mathcal{S}(i+m-1)$.
3. If the Player makes a column move from s_0 then either (1) $b(s_{0\downarrow}) = b(s_0) - 2$ and the row move to the state s_0 (when $i > 0$) is bad, or (2) $b(s_{0\downarrow}) = b(s_0)$ and the row move to the state s_0 (when $i > 0$) is good. Further, $b(s_0)$ is the number of bad states between, and including, $s_{0\downarrow}$ and $\mathcal{S}(i+m-1)$. Moreover, the Player makes the first kind of moves only if $i < m$.

Proof. Our first observation is that the counters of any state s have the same values as the counters of the state $s^{\rightarrow}$. Therefore, $b(s_{init}) = m$ since all its counters have value zero. When the Player makes a row move from the state s_0, by the above observation we have that $b(\mathcal{S}(i+1)) = b(s_0)$.

The second observation is that for any j, $0 \leq j < m$, the value of the j^{th} counter of any state s has no role in deciding whether the state is good or bad unless the index of s equals j or $j-1$ modulo m. Therefore, for $0 \leq j \leq m-2$, the state $\mathcal{S}(i+j)$ is good iff the state s_j is good (s_j is defined to be the state j columns away from s_0 in its row in the Table). The same property holds for any state $\mathcal{S}(r)$ of the Player. And so, when the Player makes a row move from s_0, it follows that the state $\mathcal{S}(i+j)$ is good iff the state s_j is good for $0 \leq j \leq m-1$, implying that $b(s_0)$ equals the number of bad states between s_0 and $\mathcal{S}(i+m-1)$.

When the Player makes a column move from s_0, it implies that the state $s_{0\downarrow}$ is good and s_0 is bad. Also, the state $\mathcal{S}(i+m-1)$ is good iff s_{m-1} is bad. Therefore, when s_{m-1} is also bad, then $b(s_{0\downarrow}) = b(s_0) - 2$, and when s_{m-1} is good then $b(s_{0\downarrow}) = b(s_0)$. Since the Player makes a column move from s_0, it must have made a row move to s_0 assuming that $i > 0$ (since $\mathcal{S}$ is a normalized strategy). If the state s_{m-1} is bad then the state ${}^{\leftarrow}\mathcal{S}(i) = \mathcal{S}(i-1)$ must also be bad since it is in the same row of the Table. Thus, the row move from ${}^{\leftarrow}\mathcal{S}(i)$ is bad iff $b(s_{0\downarrow}) = b(s_0) - 2$. Also, by Lemma 5.9, when the state ${}^{\leftarrow}\mathcal{S}(i)$ is bad, a column move is made from the state s_0 only if $i < m$. □

Lemma 5.3 For $k \geq K + 2m$, the Player makes no column moves while in segment S^4.

Proof. Suppose that the Player makes a column move from the state $\mathcal{S}(k-m+j)$ for some j, $0 \leq j < k$. Then, by the Lemma 5.9.3, the Player makes a non-reducing column move from the state $\mathcal{S}(k-m+j-r(m-1))$ for every r, $1 \leq r \leq \lfloor (k-m+j)/(m-1) \rfloor$. So, there are at least $r+1 \geq \lfloor (k-1)/(m-1) \rfloor \geq (l-1) \cdot m + 2$ non-reducing column moves in the strategy. However,

there are only $ml - 2c$ non-reducing column moves available, and—as shown in Section 5—the value of $2c$ is at least $m - 1$. This contradicts our assumption. □

Lemma 5.4 If $2c = m$ then $a' = 0$, otherwise $l - \lceil 2b/m \rceil - m + 2c \leq a' \leq l - 1$.

Proof. When $2c = m$, then, since all the reducing column moves are made in the first segment, all the states are good after $\mathcal{S}(m-1)$. Therefore, no non-reducing column moves are made, and so $a' = 0$.

Otherwise, for any non-reducing move from the state $\mathcal{S}(i)$, $i \geq m$, we have, by Lemma 5.9 that for every r, $1 \leq r \leq \lfloor i/(m-1) \rfloor$, the Player makes a column move from the state $\mathcal{S}(i - r(m-1))$. So, there is a 'chain' of non-reducing column moves starting from a state in the first segment and occurring after every $m - 1$ row moves. So, the Player will make a column move from a state in the set $\{\mathcal{S}(m + k - r'm) \mid 1 \leq r' < d - 1\}$ once for every m moves in the chain. We define the *length* of such a chain to be the number of column moves made in the chain.

Any two such chains in the Strategy must be non-overlapping, and there are exactly $m - 2c$ such chains as c reducing column moves are carried out in the first segment. Let the length of these $m - 2c$ chains be $n_1, \ldots, n_{m-2c}$. Therefore, the number a' is at least

$$\sum_{1 \leq j \leq m-2c} \lfloor n_j/m \rfloor.$$

Using the inequality $\lfloor x \rfloor + \lfloor y \rfloor \geq \lfloor x + y \rfloor - 1$, we get,

$$a' \geq \lfloor \sum_{1 \leq j \leq m-2c} n_j/m \rfloor - (m - 2c - 1).$$

Since the number of non-reducing column moves made is exactly $ml - 2b - 2c$ and $2c < m$, we get,

$$a' \geq \lfloor l - 2c/m - 2b/m \rfloor - (m - 2c - 1) \geq l - \lceil 2b/m \rceil - m + 2c.$$

The upper bound on a' simply follows from the fact that any counter value can be at most l and the chains start from the first segment which has no states in common with the third one. □

Lemma 5.5 S^2 has at least $\max\{0, \min\{m - 2c, k + 2 - dm - 2c\}\}$ bad states.

Proof. The segment S^2 has states $\mathcal{S}(m)$ to $\overleftarrow{\mathcal{S}}(m + k - dm)$. Of these, only the states $\mathcal{S}(m)$, $\mathcal{S}(m+1), \ldots, \mathcal{S}(m+k-dm-1)$ can be bad, i.e., a total of $k - dm$ states. We have another upper bound on the number of bad states, viz., $m - 2c$ since $b(\mathcal{S}(m)) = m - 2c$. Since $k - dm < m$, there can be less that $m - 2c$ bad states in S^2—some of the $m - 2c$ bad states may be from $\mathcal{S}(k - dm), \ldots, \mathcal{S}(2m - 1)$.

Since the column move from the state s_{init} is always a reducing one, the state $\mathcal{S}(m-1)$ is a good state. Now consider the state $\mathcal{S}(m-2)$. It is, naturally, a bad state. By Lemma 5.9, there is a column move from the state $\mathcal{S}(m-2)$. Therefore, the state $\overleftarrow{\mathcal{S}}(m-1)$ is a good state. Also, $b(\overleftarrow{\mathcal{S}}(m-1)) = m - 2c$, or in other words, there are $m - 2c$ bad states amongst $\mathcal{S}(m)$, $\ldots, \mathcal{S}(2m - 3)$.

The number of bad states in the second segment can now be counted. These must be at least $m - 2c - (2m - 3 - (m + k - dm - 1)) = k + 2 - dm - 2c$. Since the number must always be between 0 and $m - 2c$, we have that the number of bad states in the segment S^2 are at least $\max\{0, \min\{m - 2c, k + 2 - dm - 2c\}\}$. □