Pinpointing Computation with Modular Queries in the Boolean Hierarchy

Manindra Agrawal * Richard Beigel † Thomas Thierauf ‡

May 27, 1997

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 [HT96] 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.

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, HT96, 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].

*Department of Computer Science, Indian Institute of Technology, Kanpur 208016, India. Email: manindra@iitk.ernet.in. Work done while visiting Abteilung Theoretische Informatik, Universität Ulm, on an 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. On sabbatical leave 1996–97 at the Dept. of Computer Science, University of Maryland, College Park, MD 20742-3251, USA. Email beigel@cs.umd.edu. Supported in part by U.S. National Science Foundation grants CCR-8952528 and CCR-9415410 and by NASA (NAG52895)

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

Definition 1.1 [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 $m \geq 2$.

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

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 1.2 [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 1.2 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 [HT96] showed that in this case the evaluator gets in fact less information (unless the Boolean hierarchy collapses).

Theorem 1.3 [HT96] *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 1.2 holds, and furthermore, that the resulting class coincides with the $(2k+1)$-th level of the Boolean hierarchy.

Theorem 1.4 [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 [HT96] showed that the resulting classes are located much lower in the Boolean hierarchy than with full information.

Theorem 1.5 [HT96] *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 1.6 [HT96] *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 $\mathrm{NP}(l)//\mathrm{Mod}_m^{\mathrm{NP}[k]}$, for $l \geq 1$.

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 $\Sigma = \{0, 1\}$. For a predicate P, let $[P]$ denote 1 if P is true, and 0 if P is false.

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 $\mathrm{NP}(k)$ ($k \geq 1$), the *k-th level of the Boolean hierarchy*, if there exist $A_1, \ldots, A_k \in \mathrm{NP}$ 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).

We note that in the above definition of NP(k), we can require in addition that the sets A_i form a decreasing chain $A_1 \supseteq \cdots \supseteq A_k$ and we still get the same class, NP(k) [CGH+88].

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. Since their proof technique is the key also to our results, we include a proof of their theorem. Let us first define some notions.

Let α be a Boolean function with k variables, that is $\alpha : (x_1, \ldots, x_k) \in \{0,1\}^k \mapsto \{0,1\}$. Then NP($\alpha$) be the class of sets L that can be expressed by k sets $L_1, \ldots, L_k \in$ NP as follows. For any $x \in \Sigma^*$, set variable $x_i = 1$ if and only if input $x \in L_i$, for $i = 1, \ldots, k$. Then we must have $x \in L \iff \alpha(x_1, \ldots, x_k) = 1$.

The result of Wagner and Wechsung [Wec85] says that for any α there is some m such that NP(α) coincides with either NP(m) or coNP(m). A crucial point thereby is to determine the value of m.

Let $\mathbf{a} = (a_1, \ldots, a_t)$ be an increasing chain in $\{0,1\}^k$ with respect to the bitwise order $\prec$. That is, $a_i \in \{0,1\}^k$, for $i = 1, \ldots, t$, and $a_1 \prec \ldots \prec a_t$. 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,*

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

Proof. Let $L \in$ NP(α) via $L_1, \ldots, L_k \in$ NP. Define sets A_i, $i = 1, \ldots, m(\alpha)$, as follows

$$x \in A_i \iff \begin{array}{l} \text{there exists an increasing chain } \mathbf{a} = (a_0, \ldots, a_i) \text{ in } \{0,1\}^k \\ \text{where } \alpha \text{ makes } i \text{ mind-changes and such that} \\ \text{if there is a 1 at position } j \text{ in } a_i, \text{ then } x \in L_j. \end{array}$$

Clearly, all sets A_i are in NP. Moreover, they form a decreasing inclusion chain: $A_1 \supseteq \cdots \supseteq A_{m(\alpha)}$. Now, let $A = A_1 \triangle \cdots \triangle A_{m(\alpha)} \in$ NP($m(\alpha)$). That is, $x \in A$ if and only if the maximum i such that $x \in A_i$ is odd. In other words, $x \in A$ if and only if the maximum number of mind-changes that α can make with respect to x is odd. Therefore $L = A$ if $\alpha(0^k) = 0$ and $L = \overline{A}$ if $\alpha(0^k) = 1$.

For the reverse direction let $L = A_1 \triangle \cdots \triangle A_{m(\alpha)} \in$ NP($m(\alpha)$), for NP sets $A_1 \supseteq \cdots \supseteq A_{m(\alpha)}$. Furthermore, let $(a_0, \ldots, a_{m(\alpha)})$ be an in increasing chain in $\{0,1\}^k$ where α makes $m(\alpha)$ mind-changes. We can assume that $a_{m(\alpha)} = 1^k$.

Now we define sets $L_1, \ldots, L_k$ that define L via α. For all $j \in \{1, \ldots, k\}$, if a_1 has a 1 at position j, then $L_j = A_1$. In other words, we write A_1 in all variable positions of α where a_1 has a 1. In the next step, we write A_2 in all variable positions of α where a_2 has a 1 that is not already in a_1. That is, for all $j \in \{1, \ldots, k\}$, if a_1 has a 0 at position j and a_2 has a 1 at position j then $L_j = A_2$. Continuing that way up to $m(\alpha)$, we get k sets $L_1, \ldots, L_k$ that define L via α if $\alpha(0^k) = 0$, and that define $\overline{L}$ if $\alpha(0^k) = 1$. □

Let α be a k-ary Boolean function. An upper bound on the number of mind-changes of α is the number of variables, k, since no increasing chain in $\{0,1\}^k$ can be longer than $k+1$. Therefore, $0 \leq m(\alpha) \leq k$.

Consider for example the k-ary *parity function* par_k. The sequence with $a_i = 1^i 0^{k-i}$ for $i = 0, \ldots, k$ forms an increasing chain. Since the number of mind-changes matches the upper bound, we have $m(\mathrm{par}_k) = k$. Note that the class $\mathrm{NP}(k)$ is defined via the k-ary parity function. Since $\mathrm{par}_k(0^k) = 0$, we get back $\mathrm{NP}(k)$ by Theorem 3.1.

As a more interesting example, consider classes $\mathrm{NP}(l)//\#^{\mathrm{NP}[k]}$. The Boolean function α associated with $\mathrm{NP}(l)//\#^{\mathrm{NP}[k]}$ has $k + l(k+1)$ variables, namely, k variables $x_1, \ldots, x_k$ for the $\#^{\mathrm{NP}[k]}$ function and l variables $y_{i1}, \ldots, y_{il}$, for each potential function value $i \in \{0, \ldots, k\}$ of the $\#^{\mathrm{NP}[k]}$ function. Then the value of α is $\mathrm{par}_l(y_{i1}, \ldots, y_{il})$, if i of the x_j variables are one. Note that to evaluate α, we don't need to know the exact assignment to its variables: since 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 $i = 0, \ldots, k$. The collection of these numbers, we call a *state*.

Definition 3.2 *A* state *s is a $k+2$ tuple of integers $s = (c_0, c_1, \ldots, c_k, i)$ such that $0 \leq c_j \leq l$ for $0 \leq j \leq k$, and $0 \leq i \leq k$. We refer to the last component of s (the number i) as the* index *of s, and the numbers 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 $s = (c_0, \ldots, c_k, i)$, the index i denotes the number of ones in $(x_1, \ldots, x_k)$ and the counter c_j denotes the number of ones in $(y_{j1}, \ldots, y_{jl})$. The function α can now be thought of as acting on states: $\alpha(s) = c_i \bmod 2$.

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 3.3 *The* table *consists of $k+1$ columns and $kl+1$ rows. The entries in the table are states. For $0 \leq C \leq kl$ and $0 \leq i \leq k$, the entry at position (C, i) of the table can be any state $s = (c_0, \ldots, c_k, i)$ with $\sum_{j=0}^{k} c_j = C$.*

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

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

respectively. Similarly, the upper and lower column neighbours *of s are*

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

respectively.

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

Definition 3.4 *Let $s = (c_0, \ldots, c_k, i)$ 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 values of the active counters of the two adjacent states involved in the move have different parity (i.e., are different modulo 2), and 0 otherwise.*

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

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 3.5 *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.*

An optimal playing strategy *for the player is a strategy that has the maximum number of mind-changes.*

For a strategy $\mathcal{S}$, let $\mathcal{S}(i)$ denote the state reached by the player immediately after the i^{th} row move according to the strategy $\mathcal{S}$.

The mind-change of a column move is always one. Only for the row moves the mind-change can be zero, and therefore, these have to be made carefully in order to arrive at an optimal playing strategy.

Definition 3.6 *A state $s = (c_0, \ldots, c_k, i)$ is* good *if $c_i \not\equiv c_{i+1} \pmod 2$, and* bad *otherwise. A row move from a good (bad) state is a* good (bad) move.

Coming back to classes $\mathrm{NP}(l)//\#^{\mathrm{NP}[k]}$, it is clear that the number of mind-changes of an optimal strategy equals $m(\alpha)$. The optimal strategy is quite easy to see:

STRATEGY FOR $\mathrm{NP}(l)//\#^{\mathrm{NP}[k]}$

```
for i ← 0 to k do
    make l column moves
    if i < k then make a row move
```

At any time, variable i contains the column of the table where column moves are made. In each column, we simply make the maximum number of 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, when l is even, we get $l(k+1)$ mind-changes, and, when l is odd, we get additionally k mind-changes, i.e., $l(k+1)+k$ in total.

When l is odd, the number of mind-changes matches the upper bound, the number of variables of α. 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 Theorem 1.4.

Theorem 3.7 *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 from above, we have now only m counters $c_0, \ldots, c_{m-1}$ in a state, and, for any $i \in \{0, \ldots, k\}$, the counter $c_{i \bmod m}$ is the active counter. Hence, after every m row moves the counter c_i becomes active again, for every i, $0 \leq i < m$.

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 $c_{i \bmod m}$.

4 Strategies for $\mathrm{NP}(l)//\mathrm{Mod}_m^{\mathrm{NP}[k]}$

Let us start by naivly adapting the strategy for $\mathrm{NP}(l)//\#^{\mathrm{NP}[k]}$ to classes $\mathrm{NP}(l)//\mathrm{Mod}_m^{\mathrm{NP}[k]}$. That is, during the first m row moves we fill up all the m counters. Then all our column moves are used and all states in the remaining $k-m$ row moves will be bad. Hence, we get lm mind-changes when l is even, and $lm+m-1$ when l is odd. This seems to be a good strategy only if $k=m$ because then we make no bad moves.

For larger k, a better strategy is to spend just one column move to turn a bad state into a good state. That is, we make one column move per every row move. Hence, after the first m row moves all counters have value 1, after the second m row moves all counters have value 2, and so on. After ml row moves, all counters have value l. Except when c_{m-1} is the avtive counter, the column move turns the bad actual state into a good state, then making a good row move. Still, the remaining $k-lm$ row moves are all bad. We get $2ml-l$ mind-changes by this strategy, because on $ml-l$ of the column moves we make a good row move. Observe that we obtained roughly twice as many mind-changes as in the naive strategy from above. This is clearly a good strategy as long as k is small, e.g., $k=lm$. The case of small k is treated in detail in Section 7. For larger k we can in fact do better.

Suppose now that k is large compared to m and l (we will determine later how large precisly k should be) and consider the above strategy (the second one). The initial part, where we make all our good row moves, is small, and we spend most of the time in the second part where we make only bad row moves. Therefore it would be better if there are many good states present when the second part starts. In order to have many good states, neighbouring counters must have different parity. This motivates the following definition.

Definition 4.1 *For any state $s=(c_0,\ldots,c_{m-1},i)$, we define the* badness of s, $b(s)$, *as the number of counters c_j such that $c_j \equiv c_{(j+1) \bmod m} \pmod 2$, for $j=0,\ldots,m-1$.*

For example in the initial state s_{init}, all counters have value 0, and therefore $b(s_{init})=m$ which is clearly the maximum value for b for any state. Note also that $b(s)$ is the number of bad states the player will see during the next m row moves. In other word, for $s=(c_0,\ldots,c_{m-1},i)$ and any strategy $\mathcal{S}$, there are $b(s)$ bad states in $\{s,\mathcal{S}(i+1),\ldots,\mathcal{S}(i+m-1)\}$.

How small can b become? Consider the counters that have alternating values 0 and 1. Now we get a difference of whether m is even or odd. If m is even, we get an alternating cycle and the badness b is 0. However, if m is odd, we cannot avoid to have a pair of counters with the same parity. Therefore the smallest value for b in this case is 1. Because of this difference, we distinguish the case whether m is even or odd, and give two strategies: strategy $\mathcal{A}$ for even m and strategy $\mathcal{B}$ for odd m. We will show in Section 5 that these strategies are optimal for large enough k.

4.1 For m even

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

Strategy $\mathcal{A}$

for $i \leftarrow 0$ **to** $m-1$ **do**
    make $l-1$ column moves
    if $i \not\equiv l \pmod 2$ **then** make a column move
    make a row move

```
5    for i ← m to k − 1 do make a row move
6    if c_k = l − 1 then make column move
```

The for-loop in line 1 to 4 put alternating the value $l-1$ and l in the counters. That is, state $\mathcal{A}(m)$ has the form $\mathcal{A}(m) = (l, l-1, \ldots, l, l-1, m)$ if l is odd, and $\mathcal{A}(m) = (l-1, l, \ldots, l-1, l, m)$ if l is even. From the m row moves made so far, $m/2-1$ are bad moves. Now, we have $b(\mathcal{A}(m)) = 0$, and therefore all row moves made in line 5 are good. In total, there are $k - (m/2-1)$ good row moves.

In the first for-loop, we also make $ml - m/2$ column moves. Depending on the value of c_k, there can be one more column move in line 6. When do we get an extra column move in the last column? Let $0 \leq k' < m$ such that $k' \equiv k \pmod{m}$. Thus $c_{k'}$ is the active counter at the end and we want to know whether $c_{k'} = l-1$ before executing line 6. By the condition in line 3, this is the case when $k' \equiv l \pmod 2$. Since m is even, $k' \equiv k \pmod 2$. Hence, we get an extra column move precisely when $k \equiv l \pmod 2$.

In summery, the number of mind-changes of strategy $\mathcal{A}$ is

$$\begin{aligned} mc(\mathcal{A}) &= \left(k - (\frac{m}{2} - 1)\right) + \left(ml - \frac{m}{2}\right) + [k \equiv l \pmod 2] \\ &= k + ml - m + 1 + [k \equiv l \pmod 2]\,. \end{aligned}$$

Observe the improvement over our first approaches when k is large. Recall that the number of variables of the underlying formula, i.e., $k + ml$ in this case, is an upper bound on the number mind-changes of any formula. Hence $mc(\mathcal{A})$ is off by less then m from the upper bound. We lost these mind-changes by spending half of them to set up the first m columns, and for the other half, we did less column moves than possible. In the beginning examples, we have already seen that the player can avoid making any bad moves for small k. However, for large k, we will show that bad moves cannot be avoided and that strategy $\mathcal{A}$ is optimal.

4.2 For m odd

We will develop a strategy $\mathcal{B}$ when m is odd. As already explained before Section 4.1, the smallest badness we can achieve for a state s in this case is $b(s) = 1$. In strategy $\mathcal{A}$ we did all column moves (except may be one) during the first m row moves. Now we are more careful with our column moves since we might be able to use them to decrease the number of bad moves we have to make. Hence, we will produce a state with badness 1, which is the best we can achieve, but keeping our column moves as small as possible. That is, we start by producing alternating counter values 1, 0, 1, 0, ... If we simply continue that way for the first m columns, we will end with a 1, i.e., $c_{m-1} = 1$. Observe now that because of our column move in the first column, i.e., $c_0 = 1$, the state $\mathcal{B}(m-1)$ is good. Making a column move there, turns a good state into a bad state. Hence, this doesn't look like a good idea. Instead, we do the following: we keep $\mathcal{B}(m-1)$ as a good state and make a column move already in column $m-2$. Then $b(\mathcal{B}(m)) = 1$.

If we make only row moves from now on, as it is done in strategy $\mathcal{A}$, then one move out of every m will be a bad move. But we can do even better by using column moves: if we make a column move in the bad state, the state is turned into a good state, and the badness of this good state is still 1! So we can continue that way until our counters are filled up.

STRATEGY $\mathcal{B}$

```
1    for i ← 0 to m − 2 do
2        if i is even or i = m − 2 then make a column move
```

3   make a row move
4   **for** $i \leftarrow m-1$ **to** k **do**
5   **if** $\mathcal{B}(i)$ is bad **and** $c_i < l$ **then** make a column move
6   make a row move
7   **if** $c_k = l-1$ **then** make a column move

By the for-loop in line 1 to 3, we get $\mathcal{B}(m) = (1,0,1,0,\ldots,1,0,1,1,0,m)$ and the number of bad moves made so far is $(m-3)/2$. As the player proceeds from $\mathcal{B}(m)$, the first bad state that it encounters is $\mathcal{B}(2m-3)$. Then it makes a column move (line 5) and proceeds. So, we get

$$\begin{aligned}
\mathcal{B}(2m) &= (1,0,\ldots,1,0,2,1,0,2m),\\
\mathcal{B}(3m) &= (1,0,\ldots,1,0,1,1,2,1,0,3m),\\
\mathcal{B}(4m) &= (1,0,\ldots,1,0,2,1,2,1,0,4m),\ \ldots
\end{aligned}$$

Observe that the bad state is occurring after every $m-1$ row moves by the player. Hence, after $m(m-1)$ row moves from $\mathcal{B}(m)$ on, we get $\mathcal{B}(m^2) = (2,1,\ldots,2,1,2,2,1,m^2)$, i.e., each counter is increased by one compared to $\mathcal{B}(m)$. The same pattern of moves will now be repeated. So, after $(l-1)(m-1)m$ row moves from $\mathcal{B}(m)$ on, the state of the player will be

$$\mathcal{B}((l-1)(m-1)m+m) \quad = \quad (l,l-1,\ldots,l,l-1,l,l,l-1,(l-1)(m-1)m+m).$$

(We assume that $k \geq (l-1)(m-1)m+m$.) Now, no more column moves are made, except maybe in line 7.

Thus, for the remaining $k-((l-1)(m-1)m+m)$ row moves, the player 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 $k = dm+k'$ where $0 \leq k' < m$. Then the player will make $d-(l-1)(m-1)-1$ bad moves, and one more if $k' \geq m-2$. Since $[k' \geq m-2] = \lfloor (k+2)/m \rfloor - d$, the total number of good row moves in strategy $\mathcal{B}$ is $k-(m-3)/2-(d-(l-1)(m-1)-1+\lfloor (k+2)/m \rfloor - d)$.

The number of column moves made is exactly $ml-(m-1)/2$ excluding the (possible) column move in line 7. The latter one is done if either $k' \leq m-3$ and is odd, or $k' = m-1$. This can be expressed as $[k \not\equiv \lfloor (k+2)/m \rfloor \pmod 2]$.

Summing up the good moves of strategy $\mathcal{B}$, we get

$$\begin{aligned}
mc(\mathcal{B}) &= \left(k - \frac{m-3}{2} - (d-(l-1)(m-1)-1+\left\lfloor \frac{k+2}{m} \right\rfloor - d)\right)\\
&\quad + \left(ml - \frac{m-1}{2}\right) + \left[k \not\equiv \left\lfloor \frac{k+2}{m} \right\rfloor \pmod 2\right]\\
&= k + ml - (m-4) + \left((l-1)(m-1) - \left\lfloor \frac{k+2}{m} \right\rfloor\right) + \left[k \not\equiv \left\lfloor \frac{k+2}{m} \right\rfloor \pmod 2\right]\\
&= k + 2ml - 2m - l + 4 - \left\lfloor \frac{k+2}{m} \right\rfloor + \left[k \not\equiv \left\lfloor \frac{k+2}{m} \right\rfloor \pmod 2\right]
\end{aligned}$$

Consider the second equation. From the total number of possible moves, $k+ml$, we subtract $m-4$ for the bad moves in the beginning and the missing column moves at the end, similar as in strategy $\mathcal{A}$. Then we get one bad move per every m row moves, i.e., $\lfloor (k+2)/m \rfloor$, except for the first $(l-1)(m-1)$ number of times. Compared with strategy $\mathcal{A}$ when m is even, the number of mind-changes decreased by roughly this latter term, $\lfloor (k+2)/m \rfloor - (l-1)(m-1)$. Intuitively, it seems as we cannot avoid these additional bad moves because the badness of any state is at least 1 and we already used our column moves for this as far as possible. We show in the next section that this intuition is correct and that strategy $\mathcal{B}$ is indeed optimal for odd m and large enough k.

5 Proving the Strategies Optimal

In this section, we shall obtain an upper bound on the number of mind-changes of any optimal strategy for large enough k. The upper bound we derive will match with the mind-changes for our strategies $\mathcal{A}$ and $\mathcal{B}$, thereby proving them optimal.

In the following, let $\mathcal{S}$ be any optimal strategy. At first glance it seems very hard to to make any statements about the number of mind-changes $\mathcal{S}$ will have, because there are so many possibilities of what moves $\mathcal{S}$ can do. The most important observation here is that we can modify $\mathcal{S}$ in certain ways without decreasing the number of mind-changes. This will lead to a *normal form* of a strategy that has at least as many mind-changes as the original one. Then, counting the mind-changes of the normal form of $\mathcal{S}$ is in fact possible.

At first glance, the reader might get the impression that we are trying to count the ants in an ant-hill. However, things finally settle very neat: strategies $\mathcal{A}$ and $\mathcal{B}$ turn out to be the *unique* normalization of any optimal strategy for m even and odd, respectively.

5.1 Normalizing a 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 will rearrange the column moves in $\mathcal{S}$ according to certain rules such that the number of mind-changes does not decrease. This process we call the *normalization of* $\mathcal{S}$. We arrive at the *normal form of* $\mathcal{S}$ when no more rule is applicable. In the next section, we show how to count the mind-changes of the normalized strategy.

Let $\mathcal{S}$ be a strategy of the player. By our first rule, a normalized strategy starts with a column move.

Rule 1 (making a column move from the first state) *Suppose that the first column move of strategy $\mathcal{S}$ is in the state $\mathcal{S}(j)$, for some $j > 0$. Let $\mathcal{S}'$ be the strategy obtained from $\mathcal{S}$ by deleting the first j row moves of $\mathcal{S}$ and adding j row moves at the end.*

Then $\mathcal{S}'$ starts with a column move in state $\mathcal{S}(0)$ and $mc(\mathcal{S}') \geq mc(\mathcal{S})$, since the first j row moves of $\mathcal{S}$ are all bad.

In the normalized strategy, we try to make the column moves as soon as possible. This is ensured by the following rule.

Rule 2 (shifting column moves back) *Suppose the player make a column move from a state $s = \mathcal{S}(i)$, with $i \geq m$, and such that either the state $t = \overleftarrow{\mathcal{S}}(i-m+1)$ or the state $\overleftarrow{\mathcal{S}}(i)$ 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.*

We show that $mc(\mathcal{S}') \geq mc(\mathcal{S})$. Assume first that 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 bad state t). $\mathcal{S}'$ loses a mind-change if $\overleftarrow{\mathcal{S}}(i)$ is good, because then $\overleftarrow{\mathcal{S}'}(i)$ is bad. Now assume that the state $\overleftarrow{\mathcal{S}}(i)$ is bad. Then $\overleftarrow{\mathcal{S}'}(i)$ is a good, thus adding an extra mind-change to $\mathcal{S}'$. However, $\mathcal{S}'$ may lose a mind-change if the state t is good, since the row move from $t{\downarrow}$ becomes bad in that case. Since all other moves are not affected, this proves our claim.

The main objective of the player is to make good row moves. Therefore, it should not make column move from a good state followed by a row move as this renders the row move bad. The following rule eliminates all such moves.

Rule 3 (shifting column moves from good states) *Suppose the player makes a column move from a good state s, followed by a row move to the state $\mathcal{S}(i+1)$. Let $\mathcal{S}'$ be the strategy obtained from $\mathcal{S}$ by removing the column move from s and if $i \leq k-m$ then introducing a column move to $\mathcal{S}(i+m)$.*

Notice that there may be a conflict between the rules 2 and 3 (rule 2 may shift a column move to an earlier state while rule 3 may shift the same column move to a later state). We shall define the normalization process in such way that there is no conflict due to these rules. Also notice that when $i > k-m$, the above rule is actually reducing the number of column moves in the strategy.

We claim that $mc(\mathcal{S}') \geq mc(\mathcal{S})$. 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, $\mathcal{S}'$ loses a mind-change if the row move to the state $\mathcal{S}'(i+m)$ is a bad one when $i \leq k-m$. When $i > k-m$, $\mathcal{S}'$ certainly loses a mind-change bacause it makes one less column move than $\mathcal{S}$. Since all other moves are not affected, this proves our claim.

Any even number of consecutive column moves does not change the parity of a counter. Therefore, this has no effect on the row moves and we can shift such moves to an arbitrary place. We define to make them at the beginning, that is, during the first m row moves.

Rule 4 (shifting two consecutive column moves) *Suppose that the player makes two consecutive column moves from a state $\mathcal{S}(i)$, with $i \geq m$. Let $\mathcal{S}'$ be the strategy obtained from $\mathcal{S}$ that shifts these two moves to the state $\mathcal{S}(i-m)$.*

Now we define the normal form of a strategy.

Definition 5.1 *For a strategy $\mathcal{S}$, the* normalization *of $\mathcal{S}$ is the strategy $\mathcal{S}'$ that is obtained by shifting moves of $\mathcal{S}$ in three stages. In the first stage, column moves of $\mathcal{S}$ are shifted according to Rules 1, 2, and 4 until no move can be shifted. In the second stage, the column moves are shifted according to the rule 3 until no move can be shifted. In the third stage, column moves are shifted according to the rule 4 again until no more move can be shifted. The strategy $\mathcal{S}'$ is called a* normalized strategy*.*

First note that strategies $\mathcal{A}$ and $\mathcal{B}$ in Section 4 are both normalized. If we have any optimal strategy $\mathcal{S}$, then the normal form has the same number of mind-changes as $\mathcal{S}$. This is because we do not decrease the mind-changes when applying the above rules, and, on the other hand, the mind-changes cannot increase, since $\mathcal{S}$ is optimal.

5.2 Properties of a Normalized Strategy

In the following let $\mathcal{S}$ be an optimal normalized strategy. We list some basic properties $\mathcal{S}$.

Since our rules shift column moves upward in the table, only single column moves are made from states $\mathcal{S}(i)$ for $i \geq m$. By Rule 3, column moves from good states are only possible when the next move is a column move too, or there is no more move at all. The first case is handled by Rule 4. The latter case corresponds to the extra column move we had at the end of strategies $\mathcal{A}$ and $\mathcal{B}$. There, we argued directly about the value of the last counter, c_k, by inspecting the strategies. Now, we need a more general tool. For an optimal strategy $\mathcal{S}$, when the player arrives at state $\mathcal{S}(k)$, the counter c_k must have value $l-1$ or l (since there can be at most 1 column move). Hence, we get an extra column move when $c_k = l-1$. The following lemma characterizes this event.

Lemma 5.2 *Let $\mathcal{S}'$ be the strategy obtained from $\mathcal{S}$ by deleting the last column move from the state $\mathcal{S}(k)$, if there is any (otherwise $\mathcal{S} = \mathcal{S}'$). Then $mc(\mathcal{S}) = mc(\mathcal{S}') + [l \not\equiv mc(\mathcal{S}') \pmod 2]$.*

Proof. We show by induction on the number of moves, that when the player is in state s, then the sum of the mind-changes of all the moves up to s has the same parity as the active counter of s. This is true in the beginning, when no moves are made yet, because $c_0 = 0$. Now, assume the claim holds for s and let i be the active counter of s. If the player makes a column move to $s_{\downarrow}$, then we get a mind-change. But also c_i increases by 1. If the player makes a row move to $s^{\rightarrow}$, then we get an additional mind-change if and only if s is good, which in turn is equivalent to $c_i \not\equiv c_{i+1} \pmod 2$. Hence the claim is true after any move from s.

Now the lemma follows, since the player makes a column move from $\mathcal{S}(k)$ if the active counter of the state, c_k, equals $l-1$ modulo two. □

The above extra column move at the end and also the pairs of column moves shifted by Rule 4 have no effect on the row moves. Therefore we distinguish them from the other column moves.

Definition 5.3 *For any state $\mathcal{S}(i)$, suppose the player makes c column moves from $\mathcal{S}(i)$ before making a row move, for $c \geq 2$. If c is even, we call these column moves* ineffective, *and if c is odd, we call all but one (say, the first one) of them* ineffective. *Also the column move from state $\mathcal{S}(k)$, if there is any, is called* ineffective. *All other column moves we call* effective column moves.

Our first property is now obvious.

Property 1 *Between any two consecutive row moves, the player makes at most one effective column move. Also, it makes exactly one effective column move before the first row move and no effective column move after the last row move.*

Rule 3 and 2 restrict the effective column moves of the player as follows.

Property 2 *If the player makes an effective column move from $\mathcal{S}(i)$, then $\mathcal{S}(i)$ is bad. If $i \geq m$, then additionally ${}^{\leftarrow}\mathcal{S}(i)$ and ${}^{\leftarrow}\mathcal{S}(i-m+1)$ are good states.*

Suppose the player makes a column move from $\mathcal{S}(i)$ to $t = \mathcal{S}(i)_{\downarrow}$ and that ${}^{\leftarrow}\mathcal{S}(i)$ is good. Then ${}^{\leftarrow}t$ is bad, and therefore, $\mathcal{S}(i+m-1)$ is bad. Hence, the column move has turned a bad state, $\mathcal{S}(i)$, into a good state, but $\mathcal{S}(i+m-1)$ into a bad state. It follows that $\mathcal{S}(i)$ and $\mathcal{S}(i)_{\downarrow}$ have the same badness, $b(\mathcal{S}(i)) = b(\mathcal{S}(i)_{\downarrow})$ (recall Definition 4.1 on page 7).

Only for $i \leq m$ we can have an effective column move from $\mathcal{S}(i)$ such that ${}^{\leftarrow}\mathcal{S}(i)$ is bad. In this case, ${}^{\leftarrow}t$ is good, and hence $\mathcal{S}(i+m-1)$ is good. Therefore, we get $b(\mathcal{S}(i)_{\downarrow}) = b(\mathcal{S}(i)) - 2$.

Property 3 *Let the player make an effective column move from $s = \mathcal{S}(i)$. Then either*

1. *$b(s_{\downarrow}) = b(s) - 2$, ${}^{\leftarrow}s$ is bad (or $i = 0$), and $i < m$, or*

2. *$b(s_{\downarrow}) = b(s)$ and ${}^{\leftarrow}s$ is good (or $i = 0$).*

In the latter case, note that there are $b(s)+1$ bad states in the set $\{s, \mathcal{S}(i+1), \ldots, \mathcal{S}(i+m-1)\}$.

The above property suggests that we can divide the effective column moves made by the player into two types.

Definition 5.4 *An effective column move from a state s, we call a* reducing column move*, if $b(s_{\downarrow}) = b(s) - 2$. Otherwise $b(s_{\downarrow}) = b(s)$ and we call it a* non-reducing column move*.*

By Property 3, all reducing column moves are done during the first m row moves. Since $b(s) \leq m$ for any state s, there cannot be more than $\lfloor m/2 \rfloor$ reducing column moves. Also note that if the column move is reducing, then the row move to the state s must be bad (if it exists, i.e., if $i \geq 1$). 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.

A non-reducing column move allows the player to postpone making a bad move by $m - 1$ row moves. There can be as many non-reducing column moves as permitted by the bounds on the counters of the states. Note that we used these moves in strategy $\mathcal{B}$ in the previous section.

The next property states that non-reducing column moves occur only in *chains*.

Property 4 *If $i \geq m$ and the player makes a non-reducing column move from $s = \mathcal{S}(i)$, then the player makes a non-reducing column move from the state $\mathcal{S}(i - (m-1))$.*

Proof. By Property 2, $\overleftarrow{\mathcal{S}}(i - m + 1)$ is good and s is bad. Hence, there must be a column move from $\mathcal{S}(i - m + 1)$, which must therefore be a bad state. Furthermore, this column move must be non-reducing, since the state $\overleftarrow{\mathcal{S}}(i - m + 1)$ is good. □

Finally, we can argue on one more specific column move (besides the one from s_{init}).

Property 5 *The player makes an effective column move from the state $\mathcal{S}(m-2)$.*

Proof. Since all counters are initially 0, $\mathcal{S}(m-2)$ is a bad state. It's neighbour $\mathcal{S}(m-1)$ is good, since there is a column move at s_{init} by Rule 1. Assume that the player doesn't make an effective column move from $\mathcal{S}(m-2)$. By Lemma 5.5 below, $\mathcal{S}(m - 2 + jm)$ is a bad state and the player does not make a column move from $\mathcal{S}(m - 2 + jm)$, for $j = 1, 2, \ldots$. But this is not possible, because we would obtain a strategy better than $\mathcal{S}$ by postponing one of the ineffective column moves made from $\mathcal{S}(m-2)$ to the last m row moves. □

Lemma 5.5 *Let $\mathcal{S}(i)$ be a bad state and $\mathcal{S}(i+1)$ be good. If the player doesn't make a column move from $\mathcal{S}(i)$, then $\mathcal{S}(i + jm)$ are all bad states and the player does not make a column move from any of them, for $j = 1, 2, \ldots$.*

Proof. Assume the lemma is not true. Then there must be a smallest $j \geq 1$ such that either $\mathcal{S}(i + jm)$ is a bad state and the player makes a column move from it or the state $\mathcal{S}(i + jm)$ is a good state. In the first case, Rule 2 would have shifted the column move to $\mathcal{S}(i + (j-1)m)$, contradicting our choice of j. Therefore $\mathcal{S}(i + jm)$ must be a good state. Since there is no column move from the bad state $\mathcal{S}(i + jm - m)$, there must have been a column move from the state $\mathcal{S}(i + jm - (m-1))$. Since $\overleftarrow{\mathcal{S}}(i + jm - (m-1)) = \mathcal{S}(i + jm - m)$ is bad, we must have $i + jm - m + 1 < m$ by Property 2. It follows that $j = 1$. But $\mathcal{S}(i+1)$ is good by assumption and hence, there is no column move from $\mathcal{S}(i+1)$. □

5.3 Counting the Mind-Changes

In this section, we derive an upper bound on the number of mind-changes of an optimal normalized strategy $\mathcal{S}$. For this, we provide a lower bound on the number of bad moves of the strategy. Then we simply have to subtract it from the number of all moves made.

We start by considering how many moves are made in total. Clearly there are k row moves. Potentially, there can be up to ml column moves. Let ρ denote the number of reducing column moves of strategy $\mathcal{S}$ and $\imath$ be the (even) number of ineffective column moves done during the first m row moves. Recall that by Property 3, the player makes all the reducing column moves during the first m row moves and therefore $\rho \leq m/2$. Since $b(s_{init}) = m$, we have $b(s) \geq m - 2\rho$ for any state s, and $b(\mathcal{S}(i)) = m - 2\rho$ for all $i \geq m$.

Lemma 5.6 *The player makes $C_{\mathit{eff}} = ml - \rho - \imath$ effective column moves and $M_{total} = k + lm - \rho + [S(k)\downarrow \in \mathcal{S}]$ moves in total.*

Proof. Consider the state $S(k)$. Since $\mathcal{S}$ is optimal, all counters in state $S(k)$ have value $l-1$ or l. Because $b(\mathcal{S}(k)) = m - 2\rho$, precisely ρ counters have value $l-1$. Hence, there have been $ml - \rho$ column moves when reaching state $\mathcal{S}(k)$. All but $\imath$ many of these are effective.

For the total number of moves we have to add k row moves and a possible column move from $S(k)$. The latter is expressed by the predicate "$S(k)\downarrow \in \mathcal{S}$". □

We partition the sequence of states during the whole game into four segments, S^1, S^2, S^3 and S^4, which we consider separately. To get at least some intuition for what follows, one should keep in mind how our strategies $\mathcal{A}$ and $\mathcal{B}$ work. The bounds on the number of good, respectively, bad moves we give below will depend on certain parameters. Then observe that if one plugs in the values these parameters have in strategies $\mathcal{A}$ and $\mathcal{B}$, all bounds are in fact tight. This way we get that the strategies are optimal.

The segments are defined as follows. Let $k = dm + k'$, where $0 \leq k' < m$.

Segment	from state	to state
S^1	$\mathcal{S}(0) = s_{init}$	$\leftharpoondown\mathcal{S}(m)$
S^2	$\mathcal{S}(m)$	$\leftharpoondown\mathcal{S}(m+k')$
S^3	$\mathcal{S}(m+k')$	$\leftharpoondown\mathcal{S}(k-m)$
S^4	$\mathcal{S}(k-m)$	$\leftharpoondown\mathcal{S}(k)$

We leave out state $S(k)$ here. The only possible move from $S(k)$ is an ineffective column move which we will simply add at the end.

We start by considering S^1. Since every reducing column move must be preceded by a bad move—except when it is made from state s_{init}—the player makes $\rho - 1$ bad moves to set up these column moves.

For the remaining states of S^1, we don't count the bad moves directly. This is because we cannot say in beforehand when the player makes a column move from a bad state and when not. Instead, we count the number of bad states and subtract the number of effective column moves made from it. This gives us the number of bad moves. Since we have already counted the bad moves made just before reducing column moves, there are $m - \rho$ bad states left in this segment. To these, we add the number of bad states of segments S^2 and S^3.

Next, we consider the second segment S^2. The player makes $k' < m$ row moves while it is in the states of this segment. We give a lower bound on the number of bad states in S^2.

Lemma 5.7 *S^2 has at least* $\max\{0, \min\{m - 2\rho, k' + 2 - 2\rho\}\}$ *bad states.*

Proof. The segment S^2 has states from $\mathcal{S}(m)$ to $\leftharpoondown\mathcal{S}(m+k')$. Since $b(\mathcal{S}(m)) = m - 2\rho$, there are $m - 2\rho$ bad states among $\mathcal{S}(m)$, $\mathcal{S}(m+1)$, …, $\mathcal{S}(2m-1)$. However, $\mathcal{S}(2m-1)$ and $\mathcal{S}(2m-2)$

are good states. To see this, note that $\mathcal{S}(m-1)$ is a good state, since there is a column move from the state s_{init}. $\mathcal{S}(m-2)$ is a bad state. By Property 5, there is a column move from the state $\mathcal{S}(m-2)$. Therefore, the state $\overleftarrow{\mathcal{S}}(m-1)$ is a good state. Hence, $\mathcal{S}(2m-1)$ and $\mathcal{S}(2m-2)$ are good and thus there are $m-2\rho$ bad states among $\mathcal{S}(m)$, …, $\mathcal{S}(2m-3)$.

To get a lower bound on the number of bad states in S^2, assume that the $(2m-3)-(m+k')+1$ states $\mathcal{S}(m+k'),\ldots,\mathcal{S}(2m-3)$ are all bad. Then there remain at least

$$m-2\rho-((2m-3)-(m+k')+1) \quad = \quad k'+2-2\rho$$

bad states in S^2. Since the number must always be between 0 and $m-2\rho$, the lemma follows. $\square$

Now take the third segment S^3. It has states from $\mathcal{S}(m+k')$ to $\overleftarrow{\mathcal{S}}(k-m)$. We subdivide the segment into blocks of length m. There are precisely $d-2$ such blocks (recall that $k=dm+k'$). The first state of each block is $\mathcal{S}(k'+jm)$, for $j=1,\ldots,d-2$. Now we use Property 3: if there is a column move from the state $\mathcal{S}(k'+jm)$, for some j then the number of bad states among $\mathcal{S}(k'+jm),\ldots,\mathcal{S}(k'+jm+m-1)$ is exactly $m-2\rho+1$. On the other hand, if there is no column move, then this number is exactly $m-2\rho$. Therefore, the number of bad states in segment S^3 is exactly $(d-2)(m-2\rho)+c$, where c *is the number of column moves made from the states* $\mathcal{S}(m+k-jm)$, for $1\leq j<d-1$. The following lemma gives bounds on c.

Lemma 5.8 *If* $2\rho=m$ *then* $c=0$, *otherwise* $l-\lceil\imath/m\rceil-m+2\rho\leq c\leq l-1$.

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

Now, assume that $2\rho<m$. By Property 4, any non-reducing column move from a (bad) state in the first segment starts a chain of non-reducing column moves: one after every $m-1$ row moves. Any two such chains must be non-overlapping, and there are at most $m-2\rho$ such chains, since there are exactly $m-2\rho$ bad states in the first segment. We define the *length* of such a chain to be the number of column moves made in the chain. Let the lengths of these chains be n_1, …, $n_{m-2\rho}$ (where some n_j can be 0). Note the these are all the non-reducing column moves in the whole game. By Lemma 5.6, we have $\sum_{j=1}^{m-2\rho} n_j = C_{\mathit{eff}}-\rho = ml-2\rho-\imath$.

The player will make a column move from a state in the set $\{\,\mathcal{S}(k'+jm)\mid j=1,\ldots,d-2\,\}$ once for every m moves, for every chain. Therefore, we have

$$\begin{aligned}
c &\geq \sum_{j=1}^{m-2\rho}\lfloor n_j/m\rfloor \\
&\geq \left\lfloor \sum_{j=1}^{m-2\rho} n_j/m\right\rfloor-(m-2\rho-1) \\
&= \left\lfloor\frac{ml-2\rho-\imath}{m}\right\rfloor-(m-2\rho-1) \\
&\geq l-\left\lceil\frac{2\rho}{m}\right\rceil-\left\lceil\frac{\imath}{m}\right\rceil-(m-2\rho-1) \\
&\geq l-\left\lceil\frac{\imath}{m}\right\rceil-m+2\rho.
\end{aligned}$$

For the second inequality, we used that $\lfloor x\rfloor+\lfloor y\rfloor\geq\lfloor x+y\rfloor-1$, and for the last one, recall that we assume $2\rho<m$. This proves the lower bound. The upper bound on c 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. $\square$

Finally, we consider the segment S^4. The player makes exactly m row moves while it is in the states of S^4. Assume that $\nu \leq m - 2\rho$ of the non-reducing column moves are made in S^4. Hence, the player makes $m - 2\rho - \nu$ bad moves in this segment.

Define B to be the sum of the $m - \rho$ bad states from S^1 we have not yet counted, the lower bound on the bad states of S^2 given by Lemma 5.7, and the $(d-2)(m-2\rho)+c$ bad states of S^3. That is,

$$B \;=\; m - \rho + (d-2)(m-2\rho) + c + \max\{0, \min\{m - 2\rho, k' + 2 - 2\rho\}\}. \qquad (1)$$

The total number of bad moves in the game, M_{bad}, consists of the $\rho - 1$ bad moves from S^1, the $m - 2\rho - \nu$ bad moves from S^4, and the remaining bad moves in S^1, S^2, and S^3. The latter ones we get by subtracting the number of non-reducing column moves made in these segments, namely $C_{\mathit{eff}} - \nu$, from the number of bad states, B, if this remains positive. That is

$$\begin{aligned} M_{bad} &\geq (\rho - 1) + (m - 2\rho - \nu) + \max\{0, B - (C_{\mathit{eff}} - \nu)\} \\ &= m - \rho - 1 - \nu + \max\{0, B - (C_{\mathit{eff}} - \nu)\} \end{aligned}$$

Now we get an upper bound on the number of good moves, i.e., the number of mind-changes of strategy $\mathcal{S}$, by subtracting the lower bound on the number of bad moves M_{bad} on the number of from the total number of moves, M_{total}. Therefore,

$$\begin{aligned} mc(\mathcal{S}) &\leq M_{total} - M_{bad} \\ &\leq (k + lm - \rho + [S(k)\downarrow \in \mathcal{S}]) - (m - \rho - 1 - \nu + \max\{0, B - (C_{\mathit{eff}} - \nu)\}) \\ &= k + lm - m + 1 + \nu - \max\{0, B - lm + \imath + \rho + \nu\} + [S(k)\downarrow \in \mathcal{S}]\,. \end{aligned}$$

Suppose that $k \geq 2m$. Then $d \geq 2$, and this expression is maximized if ρ and ν are chosen as large as possible. We have the bound $0 \leq \nu \leq m - 2\rho$, and therefore $\nu + 2\rho \leq m$. However, we will choose k large enough so that the expression in the above max-term, $B - lm + \imath + \rho + \nu$, is nonnegative. Then $mc(\mathcal{S})$ is in fact independent of ν. Therefore it is enough to maximize ρ.

We have two cases: if m is even, we have $2\rho = m$ (and hence, $\nu = 0$), and and m is odd, we have $2\rho = m - 1$. In the latter case, ν could be 1. However, ν must again be 0 for large enough k.

Lemma 5.9 *Let $K = (l-1)(m-1)m + 2m$. If $k \geq K$ and $2\rho \geq m - 1$, then $\nu = 0$, i.e., the player makes no column moves in segment S^4.*

Proof. Suppose that the player makes a (non-reducing) column move from the state $\mathcal{S}(k-m+t)$, for some $t \in \{0, \ldots, m-1\}$. By Property 4, there must be a chain of column moves every $m-1$ row moves, down to segment $\mathcal{S}^1$. Hence, there must be at least $\lfloor (k - m + t)/(m-1) \rfloor + 1$ non-reducing column moves in the strategy. For $k \geq K$, we have

$$\begin{aligned} \left\lfloor \frac{k - m + t}{m - 1} \right\rfloor + 1 &\geq \left\lfloor \frac{(l-1)(m-1)m + m}{m-1} \right\rfloor + 1 \\ &\geq (l-1)m + 2. \end{aligned}$$

But the number of non-reducing column moves available is only $C_{\mathit{eff}} - \rho = ml - 2\rho - \imath \leq ml - 2\rho \leq m(l-1) + 1$, since $2\rho \geq m - 1$ by our assumption. $\square$

Hence, for $k \geq K$ we have the following bound on $mc(\mathcal{S})$:

$$mc(\mathcal{S}) \quad \leq \quad k + lm - m + 1 - \max\{0, B - lm + \imath + \rho\} + [S(k)\downarrow \in \mathcal{S}] \,. \qquad (2)$$

We treat the two cases whether m is even or odd separately.
Case 1: m is even. We choose $2\rho = m$. By Lemma 5.8, we have $c = 0$ and therefore $B = m/2$ by equation 1. Moreover, since there are no non-reducing column moves, $\imath = lm - m$. For the mind-changes of strategy S, we get

$$\begin{aligned} mc(\mathcal{S}) \quad &\leq \quad k + lm - m + 1 - \max\{0, m - lm + \imath\} + [S(k)\downarrow \in \mathcal{S}] \\ &= \quad k + lm - m + 1 + [S(k)\downarrow \in \mathcal{S}] \,. \end{aligned}$$

If $k + lm - m + 1$ would be the exact number of mind-changes of $\mathcal{S}$ without counting the last extra column move, then, by Lemma 5.2, we get $[S(k)\downarrow \in \mathcal{S}] = [l \not\equiv k + lm - m + 1 \pmod 2]$, where this is equal to $[l \equiv k \pmod 2]$, since m is even. However, even though we don't know yet whether the two predicates coincides, we can anyway replace $[S(k)\downarrow \in \mathcal{S}]$ and still have an upper bound on $mc(\mathcal{S})$. Hence, our final bound on $mc(\mathcal{S})$ in this case is

$$mc(\mathcal{S}) \quad \leq \quad k + lm - m + 1 + [l \equiv k \pmod 2] \,.$$

But this bound matches precisely the mind-changes of strategy $\mathcal{A}$ described in the previous section. Therefore, strategy $\mathcal{A}$ must be optimal for $k \geq K$.
Case 2: m is odd. Now $2\rho = m - 1$. By Lemma 5.8, our bound on a now is $l - 1 - \lceil \imath/m \rceil \leq a \leq l - 1$. For B, we get from equation 1

$$B \quad = \quad \frac{m+1}{2} + (d-2) + c + \max\{0, \min\{1, k' + 3 - m\}\}. \qquad (3)$$

It is easy to verify that $\max\{0, \min\{1, k' + 3 - m\} = \lfloor (k+2)/m \rfloor - d$. Next, consider the max-term of inequality 2 and plug in B. We get

$$\begin{aligned} \max\{0, B - lm + \imath + \rho\} \quad &= \quad \max\{0, \left(\frac{m+1}{2} + (d-2) + c + \left\lfloor \frac{k+2}{m} \right\rfloor\right) - lm + \imath + \rho\} \\ &= \quad \max\{0, m - lm - 2 + \imath + c + \left\lfloor \frac{k+2}{m} \right\rfloor\}. \end{aligned}$$

Since $k \geq K$, we have that $\lfloor (k+2)/m \rfloor \geq lm - m + 3 - l$. Together with our lower bound on c, it follows that the second argument in the max-term is nonnegative. Plugging this in equation 2, we get

$$\begin{aligned} mc(\mathcal{S}) \quad &\leq \quad (k + lm - m + 1) - (m - lm - 2 + \imath + c + \left\lfloor \frac{k+2}{m} \right\rfloor) + [S(k)\downarrow \in \mathcal{S}] \\ &= \quad k + 2lm - 2m - c + 3 - \imath - \left\lfloor \frac{k+2}{m} \right\rfloor + [S(k)\downarrow \in \mathcal{S}] \,. \qquad (4) \end{aligned}$$

By the same argument as in case 1 above, we can replace $[S(k)\downarrow \in \mathcal{S}]$ by $[l \not\equiv k + 2lm - 2m - c + 3 - \imath - \lfloor (k+2)/m \rfloor \pmod 2]$.

The latter is equal to $[l \equiv k - \lfloor (k+2)/m \rfloor - c \pmod 2]$, since $\imath$ is even. Hence, the bound on $mc(\mathcal{S})$ in equation 4 is maximized when $\imath = 0$. This implies that $c = l - 1$, and we get

$$mc(\mathcal{S}) \quad \leq \quad k + 2lm - 2m - l + 4 - \left\lfloor \frac{k+2}{m} \right\rfloor + \left[k \not\equiv \left\lfloor \frac{k+2}{m} \right\rfloor \pmod 2 \right]$$

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

6 Summarizing the Results

Since strategies $\mathcal{A}$ and $\mathcal{B}$ are optimal for $k \geq K = (l-1)(m-1)m + 2m$, the mind-changes of these strategies give us the mind-changes of the underlying Boolean formula of the classes $\mathrm{NP}(l)//\mathrm{Mod}_m^{\mathrm{NP}[k]}$. Thus we can apply Theorem 3.1.

Theorem 6.1 *For $k \geq K$, we have $\mathrm{NP}(l)//\mathrm{Mod}_m^{\mathrm{NP}[k]} = \mathrm{NP}(t)$, where*

$$t \quad = \quad \begin{cases} k + lm - m + 1 + [l \equiv k \pmod 2] & \text{if } m \text{ is even,} \\ k + 2lm - 2m - l + 4 - \left\lfloor \frac{k+2}{m} \right\rfloor + \left[k \not\equiv \left\lfloor \frac{k+2}{m} \right\rfloor \pmod 2 \right] & \text{if } m \text{ is odd.} \end{cases}$$

An interesting special case is if $l = 1$. This was solved for *even* m in [HT96]. Now, we also have the result for odd m.

Corollary 6.2 *For $k \geq 2m$, we have $\mathrm{NP}//\mathrm{Mod}_m^{\mathrm{NP}[k]} = \mathrm{NP}(t)$, where*

$$t \quad = \quad \begin{cases} k + 1 + [k \text{ odd}] & \text{if } m \text{ is even,} \\ k - \left\lfloor \frac{k+2}{m} \right\rfloor + 3 + \left[k \not\equiv \left\lfloor \frac{k+2}{m} \right\rfloor \pmod 2 \right] & \text{if } m \text{ is odd.} \end{cases}$$

Another interesting case is for $m = 2$, i.e., considering classes $\mathrm{NP}(l)//\oplus^{\mathrm{NP}[k]}$. Since $\oplus^{\mathrm{NP}[k]}$ can be seen as a (characteristic function of a) set in $\mathrm{NP}(k)$, such classes represent 2 concatenated computations in the Boolean hierarchy: an $\mathrm{NP}(l)$ set gets the result of an $\mathrm{NP}(k)$ computation.

Corollary 6.3 *For $k \geq 2(l+1)$,*

$$\mathrm{NP}(l)//\oplus^{\mathrm{NP}[k]} \quad = \quad \mathrm{NP}(k + 2l - 1 + [l \equiv k \pmod 2])$$

7 If k is small

For values of k less than $K = (l-1)(m-1)m + 2m$, the mind-change of an optimal strategy can be derived in a similar way. For $l = 1$, this was done in [HT96], and for the general case, we have:

Lemma 7.1 *For $2m \leq k \leq K$, and for any optimal strategy $\mathcal{S}$,*

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

where a is the largest number satisfying the following conditions:

- $0 \leq a \leq m - 2$,
- $a \equiv m \pmod 2$, *and*

- $a \leq m \cdot (l-1) / \lfloor k/(m-1) \rfloor$.

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.

[HT96] T. Han and T. Thierauf. *Information and Computation*, 128(2), 119-125, 1996.

[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).