Theory Comput Syst
DOI 10.1007/s00224-015-9637-1

The Computational Complexity of Iterated Elimination of Dominated Strategies

Arno Pauly[1]

Abstract The computational complexity of a variety of problems from algorithmic game theory is investigated. These are variations on the question whether a strategy in a normal form game survives iterated elimination of dominated strategies. The difficulty of the computational task depends on the notion of dominance involved, on the number of distinct payoffs and whether the game is constant-sum. Most of the open cases are fully classified, and the remaining cases are shown to be equivalent to certain questions regarding elimination orders on graphs. The classifications may serve as the basis for a discussion to what extent iterated dominance could be useful to restrict rationality for computationally bounded agents.

Keywords Computational complexity · Normal form game · Dominated strategies · Strategy elimination

1 Introduction

We consider finite two-player games in strategic form, i.e. given by a pair (A, B) of $n \times m$ payoff matrices (over the integers). The idea is that one player picks a row, the other the column. The first player then receives the payoff determined by the corresponding cell in A, while the second player receives the payoff specified by B. Now if some row always provides a better payoff for the row player than some other regardless of column players choice, it makes sense for row player to exclude it from his considerations. Because the matrices are assumed as common knowledge,

✉ Arno Pauly
Arno.Pauly@cl.cam.ac.uk

[1] Computer Laboratory, University of Cambridge, Cambridge, UK

Published online: 26 June 2015

column player can infer that row player will never choose this particular row. This in turn may entice column player to exclude one of the columns, if it is an inferior choice against all **remaining** rows. Iterated elimination of dominated strategies refers to such an elimination process, continued until no further strategies (that is rows or columns) can be removed.

For a fixed pair of matrices, we define several relations between subgames, that is pairs (I, J) with $I \subseteq \{1, \ldots, n\}$, $J \subseteq \{1, \ldots, m\}$. These relations differ by the justification needed to delete a certain strategy from the game. We are interested in the computational problem of deciding whether a specific strategy in a given game may be eliminated by iterated application of such a deletion rule. Depending on the rule, the number of distinct payoffs and whether the game is constant-sum or not, most of these problems are shown to be complete for one of the complexity classes L, NL, P or NP (w.r.t. logspace reductions). Some classifications of this type have previously been obtained in [4, 7, 13], and the cases resisting full classification are discussed in further detail in Section 4.

1.1 Definitions

The elimination rules of interest are the following:

Definition 1 (**Strict Dominance**) The notion of strict dominance is defined through:

1. $(I, J) \rightarrow_{<<} (I \setminus \{i\}, J)$, if there is an $i_0 \in I$ with $A_{i_0,j} > A_{i,j}$ for all $j \in J$.
2. $(I, J) \rightarrow_{<<} (I, J \setminus \{j\})$, if there is a $j_0 \in J$ with $B_{i,j_0} > B_{i,j}$ for all $i \in I$.

Definition 2 (**Dominance**) The notion of dominance (sometimes called *weak dominance* in the literature) is defined through:

1. $(I, J) \rightarrow_{<} (I \setminus \{i\}, J)$, if there is an $i_0 \in I$ with $A_{i_0,j} \geq A_{i,j}$ for all $j \in J$, and a $j_0 \in J$ with $A_{i_0,j_0} > A_{i,j_0}$.
2. $(I, J) \rightarrow_{<} (I, J \setminus \{j\})$, if there is a $j_0 \in J$ with $B_{i,j_0} \geq B_{i,j}$ for all $i \in I$, and an $i_0 \in I$ with $B_{i_0,j_0} > B_{i_0,j}$.

Definition 3 (**Weak Dominance**) The notion of weak dominance (sometimes called *very weak dominance* in the literature) is defined through:

1. $(I, J) \rightarrow_{\leq} (I \setminus \{i\}, J)$, if there is an $i_0 \in I \setminus \{i\}$ with $A_{i_0,j} \geq A_{i,j}$ for all $j \in J$.
2. $(I, J) \rightarrow_{\leq} (I, J \setminus \{j\})$, if there is a $j_0 \in J \setminus \{j\}$ with $B_{i,j_0} \geq B_{i,j}$ for all $i \in I$.

If, for some mode of domination $\rightarrow$, we have $(I, J) \rightarrow (I \setminus \{i\}, J)$ witnessed by i_0, we say that i is (weakly / strictly) dominated by i_0. Provided that we actually move to the subgame $(I \setminus \{i\}, J)$, we say that i is eliminated by i_0. An analogous convention is used for column player's strategies. Generally, we use $\rightarrow^*$ to denote the transitive closure of the relation $\rightarrow$.

Once a game and a mode of elimination have been specified, the goal is to find a minimal subgame, i.e. a subgame not further reducible. For all three notions, Nash equilibria of the subgame are also Nash equilibria of the original game, so iterated

strategy elimination can be regarded as a preprocessing step in the computation of Nash equilibria. For iterated elimination of strictly dominated strategies the converse is also true: Strategies used in Nash equilibria are never eliminated.

It is known that there is a unique minimal element for strict dominance ($\rightrightarrows_{<<}$) for each game with finite strategy sets, while this is not true for dominance or weak dominance[1]. For weak dominance, uniqueness up to equivalence can be recovered for zero-sum games, or more generally, for games with jointly varying payoffs, as shown in [12]. A game has jointly varying payoffs, if $A_{ij} = A_{kl} \leftrightarrow B_{ij} = B_{kl}$. A very similar condition suffices to obtain order independence up to equivalence for dominance as well [14, 15].

Instead of eliminating just one dominated strategy at each step, it is possible to eliminate all currently dominated strategies at once[2]. As argued in [11], this notion can be axiomatically justified, and it yields a unique minimal result:

Definition 4 (**Simultaneous Dominance**)
$(I, J) \rightarrow_{s<} (I \setminus K, J \setminus L)$, if

1. There are $i_k \in I$, $j_k \in J$ for each $k \in K$ with $A_{i_k,j} \geq A_{k,j}$ for all $j \in J$ and $A_{i_k,j_k} > A_{k,j_k}$.
2. There are $i_l \in I$, $j_l \in J$ for each $l \in L$ with $B_{i,j_l} \geq B_{i,l}$ for all $i \in I$ and $B_{i_l,j_l} > B_{i_l,l}$.
3. $K \subseteq I$ and $L \subseteq J$ are maximal among all subsets fulfilling 1. and 2.

Another notion we will consider is the elimination of never best responses (against pure strategies[3]). This concept belongs to the realm of rationalizability conditions as considered in general in [1]. The probably best known definitions of rationalizability are the definitions of [21] and [3]. Here a strategy will be eliminated, if it is not a best response (i.e. provides its player with the best possible payoff given the choice of the opponent) to any of the remaining strategies for the opponent. As shown in [1], there is a unique maximal reduced subgame, provided that one starts with finite strategy sets.

Definition 5 (**Elimination of never best responses**) The notion of iterated elimination of never best responses against pure strategies is defined through:

1. $(I, J) \rightarrow_{br} (I \setminus \{i\}, J)$, if for each $j \in J$ there is an $i_j \in I$ with $A_{i_j,j} > A_{i,j}$.
2. $(I, J) \rightarrow_{br} (I, J \setminus \{j\})$, if for each $i \in I$ there is a $j_i \in J$ with $B_{i,j_i} > B_{i,j}$.

[1]The claim of order invariance up to strategy permutation for weak dominance given in [13, Proposition 1] is wrong. A trivial counterexample is the game $A = (0, 1)$, $B = (0, 0)$ with the two reduced forms $A' = (0)$, $B' = (0)$ and $A'' = (1)$, $B'' = (0)$.

[2]Doing so for strictly dominated strategies does not influence the final result, while doing so for weakly dominated strategies could result in empty strategy sets.

[3]Often the concept of never best responses is considered against mixed strategies instead. Then never best responses and strictly dominated strategies would coincide (compare e.g. [16, Pages 59–60]). Elimination of never best responses against pure strategies however is a distinct concept.

There are various computational problems associated with the notions of iterated elimination of strategies. Several of them for strict dominance, dominance and weak dominance have been studied in [10], showing all of them to be NP-complete for dominance and weak dominance, and most of them to be in P for strict dominance. Here we will study decision problems similar to those considered in [13] or [7], asking whether it is possible to eliminate a certain strategy.

There are two canonic encodings of bimatrix games with natural numbers as payoffs into finite strings, one based on the unary representation of the individual payoffs, one based on the binary representation. Inclusion in a given complexity class always works for both, and as all lower bounds are achieved for games with a fixed number of distinct payoffs, the choice of representation has no impact whatsoever on the computational complexity of the problems we consider. All problems are decision problems, i.e. the output is either YES or NO, thus it suffices to state when YES is the output in order to fully define the problem.

Definition 6 STRICT has an $n \times m$ game A, B and a strategy $1 \leq i \leq n$ as input, and answers YES, iff there are I, J with $(\{1, \ldots, n\}, \{1, \ldots .m\}) \rightrightarrows_{<<} (I, J)$ and $i \notin I$.

Definition 7 DOMINANCE has an $n \times m$ game A, B and a strategy $1 \leq i \leq n$ as input, and answers YES, iff there are I, J with $(\{1, \ldots, n\}, \{1, \ldots .m\}) \rightrightarrows_{<} (I, J)$ and $i \notin I$.

Definition 8 WEAK has an $n \times m$ game A, B and a strategy $1 \leq i \leq n$ as input, and answers YES, iff there are I, J with $(\{1, \ldots, n\}, \{1, \ldots .m\}) \rightrightarrows_{\leq} (I, J)$ and $i \notin I$.

Definition 9 SIMULTANEOUS has an $n \times m$ game A, B and a strategy $1 \leq i \leq n$ as input, and answers YES, iff there are I, J with $(\{1, \ldots, n\}, \{1, \ldots .m\}) \rightrightarrows_{s<} (I, J)$ and $i \notin I$.

Definition 10 RESPONSE has an $n \times m$ game A, B and a strategy $1 \leq i \leq n$ as input, and answers YES, iff there are I, J with $(\{1, \ldots, n\}, \{1, \ldots .m\}) \rightrightarrows_{br} (I, J)$ and $i \notin I$.

There are several interesting modifications to the problems introduced above, focusing on additional properties of the game. For ELIMINATION $\in$ {STRICT, DOMINANCE, WEAK, SIMULTANEOUS, RESPONSE}, we use k-ELIMINATION to denote the restriction of the respective problem to games with at most k different payoff values, that is $\max\{|\{A_{ij} \mid i \leq n, j \leq m\}|, |\{B_{ij} \mid i \leq n, j \leq m\}|\} \leq k$. Z-ELIMINATION refers to the restriction to zero-sum (or constant sum[4]) games. k-Z-ELIMINATION is defined in the straightforward way. By explicitly

[4]It is straightforward that these cases are equivalent. Technically, our examples will be constant-sum games rather than zero-sum games.

considering the number of distinct payoffs in our completeness proofs, we demonstrate that this number is not suitable for an investigation along the lines of parameterized complexity theory [9].

1.2 Previous and New Results

In each step of any elimination process at least one strategy has to be eliminated, otherwise the elimination stops. Thus, any elimination process is of polynomial length, and can be guessed and verified in polynomial time. Therefore, all problems introduced above are trivially decidable in NP. For the notions with unique minimal result, membership in P follows with the same reasoning.

NP-completeness for 2-DOMINANCE was established in [7], which of course implies NP-completeness for k-DOMINANCE ($k > 2$) and for DOMINANCE. In [13], P-hardness[5] is shown for 6-Z-WEAK. That Z-DOMINANCE can be solved in P was shown in [4].

In the present paper, we show that k-STRICT is P-complete for $k \geq 3$, and k-Z-STRICT is P-complete for $k \geq 4$. Both 2-STRICT and 3-Z-STRICT are NL-complete. In a zero-sum game with payoffs in $\{0, 1\}$, only trivial cases of strict dominance are possible, and there cannot be any iteration of elimination. This allows to decide 2-Z-STRICT in L.

By proving them to be equivalent to 2-STRICT, also RESPONSE and k-RESPONSE ($k \geq 2$) will be shown to be NL-complete.

For 3-Z-DOMINANCE as well as for 3-Z-SIMULTANEOUS we show P-completeness, leaving the case $k = 2$ open. Without the restriction to zero-sum games, we can show that 2-SIMULTANEOUS is P-complete.

For weak dominance, we establish the membership in P of Z-WEAK, and reduce the number of payoff values needed for P-completeness, so that we can prove 3-Z-WEAK to be P-complete. The problem 3-WEAK is NP-complete. For the case of just two different payoff values, we only establish P-hardness of 2-WEAK, and leave the remaining questions open. See Table 1 on Page 5 for an overview.

2 Complexity Classes and Their Complete Problems

The computational complexity of a problem is usually classified by showing it to be among the hardest problems solvable by a specific computational model. *Hardest* refers to reducibility of problems: If there is a simple procedure f to compute an instance $f(i)$ of problem B from any instance i of problem A, such that the answer of A to i is the same as the answer of B to $f(i)$, then A is reducible to B. A problem A is *hard* for some class of problems $\mathcal{C}$, if any problem in $\mathcal{C}$ is reducible to A. If moreover A is in $\mathcal{C}$, we call it $\mathcal{C}$-complete. The specific reduction type used in the

[5]While P-completeness is claimed, the corresponding part of the proof is wrong. However, as we will show later, the P-completeness result is true.

Table 1 Overview of the classifications

	WEAK	DOMINANCE	STRICT	SIMULTENEOUS	RESPONSE
3	NP	NP	P	P	NL
2	$\geq P$	NP	NL	P	NL
4-Z	P	P	P	P	NL
3-Z	P	P	NL	P	NL
2-Z	?	?	L	?	L

An entry NP denotes NP-completeness, etc., while $\geq P$ denotes membership in NP together with P-hardness.

present paper is logspace reductions, that means we demand that the function f in a reduction can be computed using amounts of memory growing only logarithmically with the input size.

We shall briefly introduce the complexity classes occurring in our results, and present their complete problems we use to derive the completeness-results. Most of this section is based on [17].

The class NP contains the problems decidable by a polynomially time-bounded nondeterministic Turing machine. NP-complete problems are not assumed to admit fast decision algorithms. An archetypical NP-complete problem is 3SAT, defined as following:

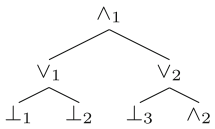
Definition 11 (**3-Satisfiability**) 3SAT has a list of clauses as input, each containing three literals of the form X_i or $\neg X_i$. The question is whether truth values can be attached to the literals, so that at least one literal per clause is true.

The problems in P are those decidable by a polynomially time-bounded deterministic Turing machine. The P-complete problems are those where a substantial speed-up through parallel computing is least to be expected, i.e. the inherently sequential problems. As an archetypical P-complete problem we use several versions of the monotone circuit value problem (MCV):

Definition 12 (**monotone circuit value problem**) The problem MCV takes a monotone circuit as input, that is a directed acyclic graph (DAG), where the vertices are labeled with AND, OR and FALSE. Vertices labeled with FALSE always have in-degree 0, while AND and OR vertices have in-degree less or equal 2. The value of a FALSE vertex is *false*. An AND vertex has the value *true*, if and only if all his input vertices have value *true*[6]; an OR vertex has value *true*, if and only if he has an input

[6]In particular, an AND vertex with in-degree 0 always has the value *true*; thus, we do not need to include designated TRUE vertices. In theory, the same is true for FALSE and OR vertices, however, including FALSE vertices explicitly facilitates our constructions.

Table 2 An example instance for MCV1 introduced in Definition 13



vertex with value *true*. There is exactly one vertex with out-degree 0, the root. The answer to the problem is *yes*, if and only if the root is assigned the value *true*.

There are two different sets of additional restrictions imposed on the circuits we will use, both lead to problems equivalent to the original MCV:

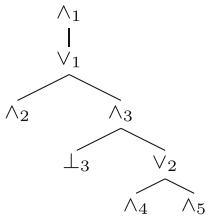
Definition 13 In the problem MCV1, AND and OR vertices are alternating. Each FALSE vertex has out-degree 1 and is input to a specific OR vertex; AND vertices only have OR vertices as input (if any). All OR vertices have in-degree 2. Two vertices never share all their inputs (except when the set is empty). The root is labeled AND. There is at least one vertex with each label (Table 2).

Definition 14 In the problem MCV2, AND and OR vertices are alternating. OR vertices only have AND vertices as input. All OR vertices have in-degree 2. Each AND-vertex has at most one FALSE-vertex as input. The root is labeled AND, and has no FALSE-vertex as input. Different AND-vertices have disjoint inputs, different OR-vertices have unequal inputs (Table 3).

The third complexity class needed is NL, the class of problems decidable on a logarithmically space-bounded nondeterministic Turing machine. The standard complete problem for NL is REACHABILITY, defined as:

Definition 15 (**Reachability**) The decision problem REACHABILITY takes a DAG G together with two vertices s, t as input, and answers YES, iff there is a path in G from s to t.

Table 3 An example instance for MCV2 introduced in Definition 14



For our purposes, another problem is more useful:

Definition 16 (**Cycle Reachability**) The decision problem CYCLEREACH takes a directed graph G and a vertex s as input, and answers YES, iff G contains a cycle which can be reached from s.

Theorem 1 CYCLEREACH *is NL-complete.*[7]

Proof A non-deterministic algorithm for CYCLEREACH storing only a constant number of vertices works as follows. Guess a node $u \in V$ which is kept for the rest of the algorithm. Starting with v as the active vertex, always guess a successor of the active vertex, and let it be the new active vertex. If there is no successor, reject. If the active vertex equals u for the first time, flip a control bit. If it equals u for the second time, accept. In addition, the number of steps can be counted, and the computation can be aborted once its exceeds $2|V|$.

To see that CYCLEREACH is even NL-complete, we present a reduction from REACHABILITY. If a path from s to t is sought, an edge from t to s is added. Then a cycle can be reached from s, iff t was reachable from s in the original graph. □

We will later use CYCLEREACH in a more restricted form, by requiring the graph to be bipartite and not to have cycles of length 2. This is of no consequence for its completeness, as we may just insert an extra vertex in the middle of any edge to ensure these properties without altering the answer.

3 The Classifications

We proceed to prove the claimed results. As the proof ideas for the membership and hardness results for any given complexity class share similar elements, we go by complexity classes, rather than by properties of the games or the dominance criteria.

3.1 Inside NL

Theorem 2 2-Z-STRICT *is in L.*

Proof The only possible dominations are by a row of 1s against a row of 0s, or by a column of 0s against a column of 1s. If we assume that the initial game allows to eliminate a row, there must be a row containing only 1s, which of course is uneliminable. Thus, there will never be a column containing only 0s, that means there will never be a domination between columns.

The considerations above show that the following algorithm is sufficient to solve 2-Z-STRICT. In the first step, determine whether the game has a row containing only 1s, or a column containing only 0s, or neither. In the first case, the initial game is

[7] I would like to thank Anuj Dawar and Yuguo He for pointing out this result to me.

copied to the output tape row-wise, leaving out all 0-rows, in the second case, it is copied column-wise, leaving out all 1-columns. In the third case, the complete game forms the output. □

Theorem 3 2-STRICT *is NL-complete.*

Proof We show that 2-STRICT and CYCLEREACH are equivalent.

Given a game, we can check in logarithmic space whether there are strategies for both row and column player always granting a payoff of 1 to the respective player after the first round of elimination. If not, the iteration stops and the answer can be determined already, as not further eliminations can be possible.

If there are such strategies, we consider the remaining strategies of both players as vertices in a graph. There is an edge from s_i to t_j, iff $A_{ij} = 1$, and an edge from t_j to s_i iff $B_{ij} = 1$. There are no edges between s_i and s_k or between t_j and t_l. Iterated elimination of strictly dominated strategies now corresponds to iteratively removing vertices without outgoing edges, as those vertices correspond to strategies always yielding payoff 0, which is then strictly dominated by the strategy always providing 1. In the end, only those vertices remain from which a cycle can be reached.

For the other direction, we start with transforming the directed graph into a bipartite graph by inserting a new vertex for each edge. Then we construct a game, where both player have a special strategy always yielding payoff 1, and a strategy for each vertex in their set of vertices. An edge from u to v corresponds to payoff 1 for the owner of u when u is played against v; no edge to payoff 0. Again, removal of dominated strategies corresponds to removal of vertices without outgoing edges. □

Theorem 4 3-Z-STRICT *is NL-complete.*

Lemma 1 3-Z-STRICT *is NL-hard.*

Proof We use a reduction from CYCLEREACH to the problem at hand. We assume the graph is bipartite and does not contain a cycle of length 2.

The Construction: The game to be constructed has fixed strategies $s_\circ$ and $t_\circ$ for row and column players respectively, and additional strategies s_u and t_v for nodes $u \in V_1$ or $v \in V_2$, if $\{V_1, V_2\}$ is the partition of the vertex set. The payoffs for row player are as follows:

	$t_\circ$	t_v
$s_\circ$	1	2
s_u	0	$\begin{cases} 2 & (u,v) \in E \\ 0 & (v,u) \in E \\ 1 & \text{otherwise} \end{cases}$

Complexity of the construction: Clearly in doable in logspace.

Correctness of the construction: A strategy t_v gets eliminated by $t_\circ$ iff v has no successors in the graph, the same holds for s_u. As $s_\circ$ and $t_\circ$ are never eliminated, there cannot be eliminations of t_{v_1} by t_{v_2} or of s_{u_1} by s_{u_2} for any vertices v_1, v_2, u_1, u_2.

What remains after iterated elimination are just the strategies belonging to those vertices forming that largest subgraph where any vertex has an outgoing edge – but these are exactly those vertices from which a cycle can be reached. □

Lemma 2 3-Z-STRICT *is in NL.*

Proof Now we have to show that iterated elimination can be executed in NL under the current restrictions. We do this by providing a logspace reduction to CYCLEREACH. As all instances where there is a row consisting only of 2s or a column consisting only of 0s are trivial, (and this property is logspace-decidable), we may freely assume that our games do not have such rows or columns.

The construction: Let R_D be the set of rows containing only 1's and 2's, and let C_D be the set of columns containing only 1's and 0's. Then let R_E be the set of rows that yield payoff 0 against all columns in C_D, and let C_E be the set of columns yielding payoff 2 against all rows in R_D.

The sets R_E and C_E are now used as vertices in a graph: There is an edge from $v \in R_E$ to $u \in C_E$, if v against u yields 2 and an edge from $u \in C_E$ to $v \in R_E$, if u against v returns 0. We add some further edges: We test for each $v \in R_E$ whether there is a $\hat{v} \in R_D$, so that $\hat{v}$ yields strictly more than v against all $u \notin C_E$. If there is no such strategy, we add a cycle and an edge from v to the cycle to our graph. The same procedure is executed for columns.

If the designated query strategy is in R_E or C_E, then the strategy can be eliminated iff **no** cycle can be reached from the corresponding vertex (note that NL = coNL). If not, then the strategy is not eliminable anyway.

Complexity of the construction: The sets R_D and C_D are clearly logspace-decidable – we just need a single bit for the answer, and then to move through a column/row in the matrix. But then also R_E and C_D are logspace-decidable, using the same linear-search argument. The final test then requires to keep track of four indices, which of course is still doable in logspace.

Correctness of the construction: We note the following properties for games containing neither a row of all 2's nor a column of all 0s:

1. A row can potentially be strictly dominated only if it contains a 0.
2. A row can potentially strictly dominate another row only if it does not contain a 0.
3. A row cannot be strictly dominated if it contains a 2.
4. A column can potentially be strictly dominated only if it contains a 2.
5. A column can potentially strictly dominate another row only if it does not contain a 2.
6. A column cannot be strictly dominated if it contains a 0.

To see that 1. is true, note that if a row x is strictly dominated by some other row y, and as assumed, y does not contain only 2s, then y has to contain a 1 somewhere, and at the corresponding place, x must have a 0. The rest are immediate, or follow by symmetry.

Combining these observations, an elimination never changes whether a row contains 0s or not, or whether a column contains 2s or not. Thus, each row (column)

is potentially strictly dominating, throughout the iterations, or never. Note that the sets R_D and C_D contain precisely these candidates. If some column c returns 0 or 1 against any row in R_D, then c cannot be eliminated, since no column can return strictly less against the same row, and the row is uneliminable. By the same argument, each potentially eliminable row must return 0 against any column from C_D. Thus, our sets R_E and C_E contain all candidates for elimination.

Next, we claim that a strategy in $R_E \cup C_E$ can be eliminated iff the corresponding vertex has no successors – thus eventually, only those strategies remain from where a cycle can be reached.

By symmetry, it suffices to consider some $v \in R_E$. Any strategy $\hat{v}$ that could eliminate v has to be an element of R_D, and has to yield strictly better payoff than v against all $u \notin C_E$ (because the $u \notin C_E$ can never be eliminated). If no such strategy exists, by construction a cycle can be reached from v. Let us assume that such a strategy exists. By construction of C_E, for any $u \in C_E$ we know that $\hat{v}$ yields payoff 2 against C_E. So $\hat{v}$ strictly dominates v iff v does not yield payoff 2 against any $u \in C_E$ – but that just is equivalent to v having no outgoing edge. □

Theorem 5 *k-RESPONSE is NL-complete for $k \geq 2$. RESPONSE is NL-complete.*

Proof Considering the trivial reducibilities between the problems concerned, it is sufficient to show that 2-REPONSE is NL-hard and that RESPONSE is in NL. For the former, note that the reduction from CYCLEREACH to 2-STRICT presented in the proof of Theorem 3 also is a reduction from CYCLEREACH to 2-RESPONSE. For the latter, see the following lemma. □

Lemma 3 RESPONSE *is in NL.*

Proof To show membership in NL for RESPONSE, we present a reduction to 2-STRICT.

The construction: Given the $(n \times m)$ payoff matrix A for row player, we construct a new matrix $\hat{A}$, where $\hat{A}_{ij} = 1$, iff $A_{ij} \geq A_{kj}$ for all k and $\hat{A}_{ij} = 0$ otherwise. Likewise for column player, the best payoffs in each row are replaced by 1, all other values by 0. We add a new strategy ($n+1$ and $m+1$ respectively) to each player, and set $A_{n+1,j} = 1$, $A_{i,m+1} = 0$ for $i \neq n+1$, $B_{i,m+1} = 1$, $B_{n+1,j} = 0$ for $j \neq m+1$.

The designated query strategy remains the same.

Complexity of the construction: As each payoff in the new game can be determined by a single linear search, this is indeed a logspace reduction.

Correctness of the construction: In the resulting game, the best-response-relationships between the original strategies are unchanged. Never-best-responses are strategies always yielding 0, and these strategies will be dominated by the new all 1-strategy. Conversely, any strategy that is sometimes a best response provides a payoff of 1 in that situation, and hence cannot be strictly dominated. □

Similar to the transfer in the preceding theorem, the reduction given in Theorem 4 also shows that 3-Z-RESPONSE is NL-hard, and the considerations in Theorem 2 showing 2-Z-STRICT to be in L also apply to 2-Z-RESPONSE.

3.2 P-completeness

We start with presenting polynomial time algorithms for the problems admitting one (regardless of the number of different payoff values), and then show hardness for the lowest k-value possible. Note that the following algorithm can be executed in polynomial time: *Search for a (weakly / strictly) dominated strategy. If there is one, eliminate it and start again. If there is none, check whether the specified strategy is still there or not.* Provided that the order of elimination is irrelevant, this algorithm solves the problems at hand. This directly leads to:

Theorem 6 STRICT *is in* P.

Theorem 7 SIMULTANEOUS *is in* P.

Proof Search only for eliminable rows at first. Once they are exhausted, search only for columns. Then always switch back and forth only if no further eliminations of the current type are possible. □

The remaining cases require more work. Basically, we will show in both cases that a certain order of elimination is sufficient to eliminate anything that could be eliminated (the order of elimination depends on the strategy one is trying to eliminate).

Theorem 8 Z-WEAK *is in* P.

Proof When the naive approach described above finds a way to eliminate the strategy s, then it will yield the right answer. The only problem occurs if the initial game (A, B) is reduced to a game $(\hat{A}, \hat{B})$, so that s is iteratively eliminable in (A, B), but not in $(\hat{A}, \hat{B})$. So we assume that there is a sequence of eliminations (x_i, y_i), so that in the ith step the strategy x_i weakly dominates y_i, and by that eliminating the latter; the sequence shall end with $y_{i_{\max}} = s$. The case we have to consider is another possible elimination (x, y), that makes one of the eliminations in our sequence impossible. We denote the first elimination being made impossible by (x_k, y_k).

The only way that (x_k, y_k) is made impossible is by elimination of x_k, so we know $y = x_k$. If $x \neq y_k$, then (x_k, y_k) could simply be replaced by (x, y_k)[8]. Thus, only the situation $x_k = y$, $y_k = x$ is problematic. In this case, however, x_k and y_k have to be identical once the kth stage has been reached. Thus, elimination of y_k alone does not enable new weak dominations in later stages. Therefore, the only problematic case is $y_k = s$. Clearly, this can be avoided if we never use s as a weakly dominating strategy for eliminations.

[8]Or, if x is subsequently eliminated by another strategy z, by (z, y_k), and so on.

The question that remains is whether it might be necessary to eliminate a strategy *by* s to allow later elimination *of* s. This means the following situation:

	u	v	...
s	a	b	...
r	c	d	...
t	e	f	...
...	...	...	...

1. s weakly dominates r: $a \geq c, b \geq d$
2. u does not weakly dominate v, but will do so once r has been eliminated: $a \leq b$, $c > d, e \leq f$
3. t does not weakly dominate s, but will do so once v has been eliminated: $e \geq a$, $f < b$
4. t does not weakly dominate r: $e < c \vee f < d$

We have $f \geq e \geq a \geq c > d$, rendering both $f < d$ and $e < c$ impossible. Thus, t could be used to eliminate r in such a situation. For more steps between the elimination of r and the elimination of s, the same considerations apply; showing that elimination by s is never necessary to allow elimination of s.

Therefore, the following modification of the algorithm above solves Z-STRICT: *Search for a strategy weakly dominated by another strategy not equal to the specified strategy. If there is one, eliminate it and start again. If there is none, check whether the specified strategy is still there or not.* □

Theorem 9 3-Z-WEAK, 3-Z-DOMINANCE *and* 3-Z-SIMULTANEOUS *are P-hard.*

Proof The claims will be proven using a reduction from MCV1. From such a circuit, we will construct a zero-sum game.

The construction: Row player has a strategy $s_{\wedge n}$ for each AND vertex with number n, a strategy $s_{\bot m}$ for each FALSE vertex m, and another strategy s_B. Column player has the strategies $t_{\wedge i}$, $t_{\vee j}$ and $t_{\bot k}$ for AND vertices $\wedge_i$, OR vertices $\vee_j$, and FALSE vertices $\bot_k$.

	$t_{\wedge i}$	$t_{\bot k}$	$t_{\vee j}$
s_B	0	0	0
$s_{\wedge n}$	$\begin{cases} -1 & \text{if } i = n \\ 0 & \text{otherwise} \end{cases}$	0	$\begin{cases} 1 & \text{if } \vee_j \text{ is an input for } \wedge_n \\ -1 & \text{if } \wedge_n \text{ is an input for } \vee_j \\ 0 & \text{otherwise} \end{cases}$
$s_{\bot m}$	1	$\begin{cases} -1 & \text{if } k = m \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} -1 & \text{if } \bot_m \text{ is an input for } \vee_j \\ 0 & \text{otherwise} \end{cases}$

The payoffs of the row player are given by the table above, the column player just tries to minimize the payoffs. The elimination of a strategy $s_{\wedge n}$ (and $t_{\wedge n}$) or $t_{\vee j}$ corresponds to assigning the value *true* to the corresponding vertices n or j. Hence, the value of the circuit can be determined by asking whether the strategy corresponding to its root can be eliminated.

Complexity of the construction: As only local information is required to determine the payoffs in the game, it is easily computed in logspace from the circuit.

Correctness of the construction: We will now study which strategies can be removed under which conditions, and see that the strategies s_B, $s_{\perp m}$ and $t_{\perp k}$ could only be eliminated, if all strategies $s_{\wedge n}$ have been eliminated first, in which case the answer is already known to be *true* anyway. With these strategies remaining, elimination for $s_{\wedge n}$ (and $t_{\wedge n}$) or $t_{\vee j}$ is consistent with truth in the circuit.

(**Strategy** s_B) The strategy s_B can be weakly dominated by a strategy $s_{\wedge n}$ only if $t_{\wedge n}$ has been eliminated. In this case, $s_{\wedge n}$ was eliminated first. s_B can be weakly dominated by a strategy $s_{\perp m}$ only if $t_{\perp m}$ was eliminated first.

(**Strategy** $s_{\wedge n}$) The strategy $s_{\wedge n}$ is weakly dominated by s_B, if all strategies $t_{\vee j}$ where $\vee_j$ is an input to $\wedge_n$ have been eliminated. Provided that $t_{\wedge n}$ has not been eliminated yet (the contrary would be impossible), in this case $s_{\wedge n}$ is also dominated by s_B. $s_{\wedge n}$ cannot be weakly dominated by a strategy $s_{\wedge n'}$, as long as $t_{\wedge n'}$ is still present, and it cannot be weakly dominated by a strategy $s_{\perp m}$, as long as $t_{\perp m}$ is still present.

(**Strategy** $s_{\perp m}$) The strategy $s_{\perp m}$ cannot be weakly dominated by $s_{\wedge n}$, as long as $t_{\wedge n}$ is still present. It cannot weakly dominated by s_B, as long as a strategy $t_{\wedge i}$ is present. Weak domination by a strategy $s_{\perp m'}$ is impossible as long as $t_{\perp m'}$ is present.

(**Strategy** $t_{\wedge i}$) The strategy $t_{\wedge i}$ cannot be weakly dominated by another strategy $t_{\wedge i'}$ or by a strategy $t_{\perp k}$, as long as $s_{\wedge i}$ is present. If $s_{\wedge i}$ is eliminated, $t_{\wedge i}$ is weakly dominated by any $t_{\perp k}$. $t_{\wedge i}$ cannot be weakly dominated by a strategy $t_{\vee j}$, as long as there is a strategy $s_{\wedge n}$, such that the vertex $\vee_j$ is an input to $\wedge_n$. The existence of such a vertex can be assumed, since the root is labeled AND, and cannot be eliminated unless all inputs are eliminated first.

(**Strategy** $t_{\perp k}$) The strategy $t_{\perp k}$ cannot be eliminated by a strategy $t_{\wedge i}$ or $t_{\perp k'}$, as long as $s_{\perp k}$ is still present. Weak dominance by $t_{\vee j}$ would only be possible, if no $s_{\wedge n}$ with $\vee_j$ being an input to $\wedge_n$ would exists, as explained above, this does not happen.

(**Strategy** $t_{\vee j}$) The strategy $t_{\vee j}$ is dominated by $t_{\wedge i}$ or $t_{\perp k}$, if $\wedge_i$ or $\perp_k$ are the only remaining input to $\vee_j$, which requires the second input to be true. $t_{\vee j}$ could only be eliminated by $t_{\vee j'}$, if $\vee_j$ and $\vee_{j'}$ had the same inputs, which was ruled out in our convention, or if $t_{\vee j}$ could also have been eliminated by certain $t_{\wedge i}$ or $t_{\perp k}$. □

Example: Table 4

Theorem 10 3-STRICT *is* P-*hard.*

Proof Again a reduction from MCV1 is given.

The construction: The players have the same strategies as in the proof of Theorem 9, except for s_B, which is no longer needed. Again, elimination of a strategy corresponds to assigning *true* to the matching vertex, and eliminability

Table 4 The construction from Theorem 9 applied to the example in Table 2

	$t_{\wedge_1}$	$t_{\wedge_2}$	$t_{\perp 1}$	$t_{\perp 2}$	$t_{\perp 3}$	$t_{\vee 1}$	$t_{\vee 2}$
s_B	0	0	0	0	0	0	0
$s_{\wedge_1}$	−1	0	0	0	0	1	1
$s_{\wedge_2}$	0	−1	0	0	0	0	−1
$s_{\perp 1}$	1	1	−1	0	0	−1	0
$s_{\perp 2}$	1	1	0	−1	0	−1	0
$s_{\perp 3}$	1	1	0	0	−1	0	−1

of the strategy of the root of the circuit tells us its value. The payoffs are given by the following tables, with row players payoffs first:

	$t_{\wedge i}$	$t_{\perp k}$	$t_{\vee j}$
$s_{\wedge n}$	0	0	$\begin{cases} 1 & \text{if } \vee_j \text{ is an input for } \wedge_n \\ 0 & \text{otherwise} \end{cases}$
$s_{\perp m}$	1	1	1

	$t_{\wedge i}$	$t_{\perp k}$	$t_{\vee j}$
$s_{\wedge n}$	$\begin{cases} 1 & \text{if } i = n \\ 0 & \text{otherwise} \end{cases}$	0	$\begin{cases} 0 & \text{if } \wedge_n \text{ is an input for } \vee_j \\ -1 & \text{otherwise} \end{cases}$
$s_{\perp m}$	0	$\begin{cases} 1 & \text{if } k = m \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} 0 & \text{if } \perp_m \text{ is an input for } \vee_j \\ -1 & \text{otherwise} \end{cases}$

Complexity of the construction: As only local information is required to determine the payoffs in the game, it is easily computed in logspace from the circuit.

Table 5 The construction from Theorem 10 applied to the example in Table 2

	$t_{\wedge_1}$	$t_{\wedge_2}$	$t_{\perp 1}$	$t_{\perp 2}$	$t_{\perp 3}$	$t_{\vee 1}$	$t_{\vee 2}$
$s_{\wedge_1}$	0	0	0	0	0	1	1
$s_{\wedge_2}$	0	0	0	0	0	0	0
$s_{\perp 1}$	1	1	1	1	1	1	
$s_{\perp 2}$	1	1	1	1	1	1	
$s_{\perp 3}$	1	1	1	1	1	1	
$s_{\wedge_1}$	1	0	0	0	−1	−1	
$s_{\wedge_2}$	0	1	0	0	−1	0	
$s_{\perp 1}$	0	0	1	0	0	0	−1
$s_{\perp 2}$	0	0	0	1	0	0	−1
$s_{\perp 3}$	0	0	0	0	1	−1	0

Table 6 The construction from Theorem 11 applied to the example in Table 2

	$t_{\vee 1}$	$t_{\vee 2}$	$t_{\vee 2-2}$
s_B	3	3	2
$s_{\wedge_1}$	3	3	1
$s_{\wedge_2}$	1	1	1
$s_{\vee 1}$	2	3	2
$s_{\vee 2}$	3	2	1

Correctness of the construction: It is trivial to see that $s_{\perp m}$ can never be eliminated, and that a strategy $s_{\wedge n}$ is strictly dominated by any strategy $s_{\perp m}$, as soon as all strategies $t_{\vee j}$ corresponding to its input vertices have been removed. Thus, the removal of row players strategies corresponds exactly to the corresponding vertices being assigned the value *true*.

For column player, a strategy $t_{\wedge i}$ can never be strictly dominated as long as any strategy $s_{\wedge n}$ is still present. As the strategy $s_{\perp m}$ will never be eliminated, elimination of $t_{\perp m}$ is also impossible. The strategy $t_{\vee j}$ can be eliminated by any strategy, if there are no strategies $s_{\wedge n}$ or $s_{\perp m}$ corresponding to input vertices left, but it can also be eliminated by the strategy $t_{\wedge i}$ or $t_{\perp k}$, where $\wedge_i$ or $\perp_k$ is the only remaining strategy corresponding to an input vertex of $\vee_j$ for which $s_{\wedge i}$ or $s_{\perp k}$ is still present. Thus, for $t_{\vee j}$ to be removed, at least one of its input vertices needs to be removed earlier.

Ultimately, elimination of the strategies $s_{\wedge n}$, $t_{\wedge i}$ and $t_{\vee j}$ follows exactly the same rules as assigning *true* to the corresponding vertices in the circuit. In particular, the value of the circuit can be found by asking for eliminability of the strategy associated with its root. □

Example: Table 5

Theorem 11 4-Z-STRICT *is* P-*hard.*

Table 7 The construction from Theorem 12 applied to the example in Table 3

	$t_{\vee 1}$	$t_{\vee 2}$	$t_{\wedge\perp 1}$	$t_{\wedge\perp 2}$	$t_{\wedge\perp 3}$	$t_{\wedge\perp 4}$	$t_{\wedge\perp 5}$
$s_{\wedge_1}$	1	0	0	0	0	0	0
$s_{\wedge_2}$	0	0	0	0	0	0	0
$s_{\wedge_3}$	0	1	0	0	1	0	0
$s_{\wedge_4}$	0	0	0	0	0	0	0
$s_{\wedge_5}$	0	0	0	0	0	0	0
s_B	0	0	1	1	0	1	1
$s_{\wedge_1}$	0	0	1	0	0	0	0
$s_{\wedge_2}$	1	0	0	1	0	0	0
$s_{\wedge_3}$	1	0	0	0	1	0	0
$s_{\wedge_4}$	0	1	0	0	0	1	0
$s_{\wedge_5}$	0	1	0	0	0	0	1
s_B	0	0	1	1	1	1	1

Proof Once more a reduction from MCV1 is used; in addition to the conditions listed in Definition 13, we assume that there are at least 2 OR vertices.

The construction: Row player has a fixed strategy s_B, a strategy $s_{\wedge i}$ for each AND-vertex $\wedge_i$ and a strategy $s_{\vee n}$ for each OR-vertex $\vee_n$. Column player has strategies $t_{\vee j}$, $t_{\vee j-1}$ and $t_{\vee j-2}$ for each OR-vertex $\vee_j$, where $t_{\vee j-x}$ is only present iff the xth input to $\vee_j$ is an AND-vertex. The payoffs are given by the following table:

	$t_{\vee j}$	$t_{\vee j-x}$
s_B	3	2
$s_{\wedge i}$	$\begin{cases} 3 & \text{if } \vee_j \text{ is an input for } \wedge_i \\ 1 & \text{otherwise} \end{cases}$	$\begin{cases} 1 & \text{if } \wedge_i \text{ is the } x\text{th input to } \vee_j \\ 1 & \text{if } \vee_j \text{ is an input to } \wedge_i \\ 0 & \text{otherwise} \end{cases}$
$s_{\vee_n}$	$\begin{cases} 2 & n = j \\ 3 & \text{otherwise} \end{cases}$	$\begin{cases} 1 & n = j \\ 2 & \text{otherwise} \end{cases}$

The value of the circuit is *true*, iff the strategy corresponding to its root is eliminated.

Complexity of the construction: Once more, the construction is purely local, and thus can be executed easily in logspace.

Correctness of the construction: The only strategies that are potentially eliminable are $s_{\wedge_i}$ and $t_{\vee_j}$, corresponding to the respective vertices being assigned the value *true*. Asking whether the strategy corresponding to the root of the circuit can be eliminated provides the value of the circuit. As explained below, the other strategies are uneliminable.

(**Strategy** s_B) It is obvious that s_B can never be eliminated, since it is a best response against any of column player's strategies.

(**Strategy** $s_{\wedge i}$) If all strategies $t_{\vee j}$ where $\vee_j$ is an input vertex to $\wedge_i$ have been eliminated, the strategy $s_{\wedge i}$ will also be eliminated. If any of these strategies is still present, $s_{\wedge i}$ cannot be removed, since it achieves the maximal payoff against it.

(**Strategy** $s_{\vee n}$) The strategies $s_{\vee n}$ are uneliminable. Due to convention, there is an OR-vertex $\vee_j$ with $n \neq j$. If the vertex $\vee_j$ has at least one AND-vertex as input, there is an (uneliminable) strategy $t_{\vee j-x}$ against which $s_{\vee n}$ yields 2, which cannot be exceeded. If both inputs of $\vee_j$ are FALSE-vertices, $t_{\vee j}$ itself is uneliminable, and takes the place of $t_{\vee j-x}$ in the argument above.

(**Strategy** $t_{\vee j}$) As $s_{\vee j}$ cannot be eliminated, a strategy $t_{\vee j}$ can only be eliminated by a strategy $t_{\vee j-x}$. This happens, if and only if the strategy $s_{\wedge i}$ corresponding to the AND-vertex forming the xth input of $\vee_j$ was eliminated previously.

(**Strategy** $t_{\vee j-x}$) The strategies $t_{\vee j-x}$ can never be strictly dominated in a subgame where s_B is still present. Since s_B can never be eliminated, the same is true for all strategies $t_{\vee j-x}$. □

Example: Table 6

Theorem 12 2-WEAK, 2-DOMINANCE *and* 2-SIMULTANEUOUS *are* P*-hard.*

Proof This time, we present a reduction from MCV2. The reduction works for all three problems simultaneously, as they yield identical answers for the constructed game.

The construction: Row player has a strategy $s_{\wedge i}$ for each AND vertex $\wedge_i$, and a strategy s_B. Column player has a strategy $t_{\vee j}$ for each OR vertex $\vee_j$, and strategies $t_{\wedge \perp k}$, where k simultaneously enumerates AND and FALSE vertices. The AND-vertex k is referring to is $\wedge_k$, the corresponding FALSE-vertex is $\perp_k$. We require $\perp_k$ to be an input of $\wedge_k$, thus, if the root is $\wedge_r$, $\perp_r$ does not exist. The payoffs are given by the following tables, starting with row player:

	$t_{\vee j}$	$t_{\wedge \perp k}$
$s_{\wedge i}$	$\begin{cases} 1 & \text{if } \vee_j \text{ is input to } \wedge_i \\ 0 & \text{else} \end{cases}$	$\begin{cases} 1 & \text{if } \perp_k \text{ is input to } \wedge_i \\ 0 & \text{else} \end{cases}$
s_B	0	$\begin{cases} 1 & \text{if } \perp_k \text{ does not exists} \\ 0 & \text{else} \end{cases}$

	$t_{\vee j}$	$t_{\wedge \perp k}$
$s_{\wedge i}$	$\begin{cases} 1 & \text{if } \wedge_i \text{ is input to } \vee_j \\ 0 & \text{else} \end{cases}$	$\begin{cases} 1 & i = k \\ 0 & \text{else} \end{cases}$
s_B	0	1

The value of the circuit is *true* iff the strategy corresponding to the root is eventually eliminated.

Complexity of the reduction: As $\perp_k$ is required to be child of $\wedge_k$, even the combined enumeration is purely local. The rest of the construction is clearly doable in logspace.

Correctness of the reduction: For the following considerations, we always assume that the strategy corresponding to the root of the circuit has not been eliminated yet. Otherwise, the answer is already determined as being *true*, and further elimination do not matter.

(**Strategy** $s_{\wedge i}$) The strategy $s_{\wedge i}$ is weakly dominated by a strategy $s_{\wedge i'}$, if and only if the inputs of $\wedge_i$ are a subset of the inputs of $\wedge_{i'}$, which can only happen if the input set of $s_{\wedge i}$ is empty. If $\wedge_i$ has a FALSE-vertex as input, this is $\perp_i$ as input. Then $s_{\wedge i}$ cannot be eliminated by s_B, as long as $t_{\wedge \perp i}$ is still present. Since $t_{\wedge \perp i}$ cannot be eliminated as long as $s_{\wedge i}$ is present, this renders $s_{\wedge i}$ uneliminable.

If all existing inputs of $\wedge_i$ are OR-vertices, then $s_{\wedge i}$ is weakly dominated by s_B, once all strategies $t_{\vee j}$ corresponding to inputs of $\wedge_i$ have been removed. Since there is a strategy $t_{\wedge \perp k}$, where $\perp_k$ does not exist, in the case of weak dominance, we also have dominance. Thus, elimination of $s_{\wedge i}$ corresponds to the vertex $\wedge_i$ being assigned the value *true*.

(**Strategy** s_B) By convention, for the strategy $t_{\wedge \perp r}$ corresponding to the root, $\perp_r$ does not exist. As we assume this strategy not to be elimianted yet, and s_B is the only strategy achieving payoff 1 against it (since a non-existing vertex cannot be the input to another vertex). Thus, s_B is uneliminable.

(**Strategy** $t_{\vee j}$) Due to our convention, each vertex $\vee_j$ has exactly two AND-vertices as input. If the strategy $s_{\wedge i}$ corresponding to one of them is eliminated, then also the strategy $t_{\vee j}$ is eliminated, e.g. by $t_{\wedge \bot k}$, where $\wedge_k$ is a remaining input, or by any other strategy, if there is no remaining input. Due to the strategy s_B, weak dominance will already imply dominance.

If both of the strategies $s_{\wedge i}, s_{\wedge i'}$ corresponding to inputs of $\vee_j$ remain, then $t_{\vee j}$ cannot be eliminated. Thus, elimination of $t_{\wedge j}$ corresponds to assigning the value *true* to the vertex $\wedge_j$.

(**Strategy** $t_{\wedge \bot k}$) As long as s_B is not eliminated, a strategy $t_{\wedge \bot k}$ can never be eliminated by a strategy $t_{\vee j}$. The strategy $t_{\wedge \bot k}$ might be eliminated by another strategy $t_{\wedge \bot k'}$, only once $s_{\wedge k}$ is eliminated first. In this case, the strategy $t_{\wedge \bot k}$ has no further relevance anyway. □

Example: Table 7

3.3 NP-completeness

Theorem 13 3-WEAK *is NP-complete.*

Proof Membership in NP is straightforward: A sequence of eliminations can only be of polynomial length, so it can be guessed. Verifying whether any putative elimination is legal requires a polynomial number of comparisons, yielding overall an NP-algorithm.

A reduction from $3SAT$ shall be presented. We assume w.l.o.g. that for different clauses c and c', the set of literals occurring in c is never a subset of the set of literals occurring in c'. For a clause c, c_i refers to the ith literal in c.

The construction: Row player has a strategy s, as well as strategies s_d for each clause d and s_{l+} and s_{l-} for each variable l. Column player has a strategies t_c, t_c^i for each clause c and $i \in \{1, 2, 3\}$, as well as a strategy t_k for each variable k. The payoffs are given in Table 8 on Page 20. The first matrix contains the payoffs for the row player, the second one column player's payoffs. The formula has a satisfying assignment iff the strategy s is eventually eliminable.

Complexity of the construction: The construction is once more purely local, no non-trivial memory is required.

Correctness of the construction: We claim that the strategy s is eventually eliminable, if and only if there is a satisfying truth assignment of the original formula. For the first direction, assume that a satisfying truth assignment is given. For each variable l, the strategies s_{l+} and s_{l-} weakly dominate each other. Thus, we can eliminate s_{l+}, if the variable l is assigned the value *true*, and s_{l-} otherwise.

In the next step we will eliminate all strategies t_d. For each clause d, one of its literals must be true. Assume that the ith literal for some given clause d is true. We claim that t_d is now weakly dominated by t_d^i. t_d^i obviously provides better or equal payoff against s, all strategies s_c and all strategies s_{l+} where the literal l does not occur in d, as well as all strategies s_{l-} where the literal $\neg l$ does not occur in d. If the literal l ($\neg l$) does occur in d, but not on the ith position, both t_d^i and t_d give payoff 1 against s_{l+} (against s_{l-}). If the literal l ($\neg l$) occurs on the ith position in

Table 8 The construction used in the proof of Theorem 13

	t_c^1	t_c^2	t_c^3	t_c	t_k
s	0	0	0	2	0
s_d	$\begin{cases}1 & c=d\\ 0 & \text{else}\end{cases}$	$\begin{cases}1 & c=d\\ 0 & \text{else}\end{cases}$	$\begin{cases}1 & c=d\\ 0 & \text{else}\end{cases}$	$\begin{cases}1 & c=d\\ 0 & \text{else}\end{cases}$	0
s_{l+}	0	0	0	0	$\begin{cases}1 & l=k\\ 0 & \text{else}\end{cases}$
s_{l-}	0	0	0	0	$\begin{cases}1 & l=k\\ 0 & \text{else}\end{cases}$

	t_c^1	t_c^2	t_c^3	t_c	t_k
s	0	0	0	0	1
s_d	$\begin{cases}2 & c=d\\ 0 & \text{else}\end{cases}$	$\begin{cases}2 & c=d\\ 0 & \text{else}\end{cases}$	$\begin{cases}2 & c=d\\ 0 & \text{else}\end{cases}$	$\begin{cases}1 & c=d\\ 0 & \text{else}\end{cases}$	0
s_{l+}	$\begin{cases}1 & c_2=l\vee c_3=l\\ 0 & \text{else}\end{cases}$	$\begin{cases}1 & c_1=l\vee c_3=l\\ 0 & \text{else}\end{cases}$	$\begin{cases}1 & c_1=l\vee c_2=l\\ 0 & \text{else}\end{cases}$	$\begin{cases}1 & l\in c\\ 0 & \text{else}\end{cases}$	$\begin{cases}1 & l=k\\ 0 & \text{else}\end{cases}$
s_{l-}	$\begin{cases}1 & c_2=\neg l\vee c_3=\neg l\\ 0 & \text{else}\end{cases}$	$\begin{cases}1 & c_1=\neg l\vee c_3=\neg l\\ 0 & \text{else}\end{cases}$	$\begin{cases}1 & c_1=\neg l\vee c_2=\neg l\\ 0 & \text{else}\end{cases}$	$\begin{cases}1 & \neg l\in c\\ 0 & \text{else}\end{cases}$	$\begin{cases}1 & l=k\\ 0 & \text{else}\end{cases}$

d, then, by assumption, l is *true* (*false*), thus, the problematic strategy s_{l+} (s_{l-}) has been removed in the first step. Thus, we have covered all cases, and shown that t_d is indeed always eliminable, provided that there is a satisfying truth assignment for the formula.

Once all strategies t_d are removed, the strategy s is weakly dominated by every remaining strategy, thus, it can be removed.

For the other direction, we have to show that if s can be removed, there has to be a satisfying truth assignment for the formula. We will assume that the elimination process was stopped immediately after the removal of s.

In the first step, we will show that for each variable l, only one of the strategies s_{l+} and s_{l-} was eliminated. For that, we observe that as long as t_l is present, the two strategies are the only ones weakly dominating the other one, thus, only one of them can be removed prior to the removal of t_l. Now as long as s is present, a strategy t_l might only be weakly dominated by a strategy t_k. However, as either s_{l+} or s_{l-} is still present at this point, t_k does not weakly dominate t_l for $l \neq k$. Whether s_{l+} or s_{l-} was eliminated determines the truth value assigned to the variable l. If neither was eliminated, the truth value can be chosen arbitrarily.

As s is the only strategy providing row player with a payoff of 2 against any strategy t_d, all the strategies t_d were eliminated prior to s. We claim that t_d must have been weakly dominated by some strategy t_d^i. This is obviously true, provided that s_d was not removed previously. Now s_d can only be weakly dominated, if all strategies t_d^j for $j \in \{1, 2, 3\}$ are removed first, which in turn would require removal of s_d, showing that s_d will not be eliminated at all. Therefore, there must be an i, such that t_d was weakly dominated by t_d^i. Assume that the ith literal in d was l ($\neg l$). Then t_d^i gives less payoff to column player against s_{l+} (against s_{l-}) than t_d, thus, s_{l+} (s_{l-}) was

[10]This example was kindly provided by a referee for an earlier version of this article.

Table 9 An example[10] for the construction in Table 8 instantiated with $(p \vee q \vee \neg r) \wedge (\neg p \vee q \vee r)$

	$t^1_{c_1}$	$t^2_{c_1}$	$t^3_{c_1}$	t_{c_1}	$t^1_{c_2}$	$t^2_{c_2}$	$t^3_{c_2}$	t_{c_2}	t_p	t_q	t_r
s	(0,0)	(0,0)	(0,0)	(2,0)	(0,0)	(0,0)	(0,0)	(2,0)	(0,1)	(0,1)	(0,1)
s_{c_1}	(1,2)	(1,2)	(1,2)	(1,1)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)
s_{c_2}	(0,0)	(0,0)	(0,0)	(0,0)	(1,2)	(1,2)	(1,2)	(1,1)	(0,0)	(0,0)	(0,0)
s_{p+}	(0,0)	(0,1)	(0,1)	(0,1)	(0,0)	(0,0)	(0,0)	(0,0)	(1,1)	(0,0)	(0,0)
s_{p-}	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,1)	(0,1)	(0,1)	(1,1)	(0,0)	(0,0)
s_{q+}	(0,1)	(0,0)	(0,1)	(0,1)	(0,1)	(0,0)	(0,1)	(0,1)	(0,0)	(1,1)	(0,0)
s_{q-}	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(1,1)	(0,0)
s_{r+}	(0,0)	(0,0)	(0,0)	(0,0)	(0,1)	(0,1)	(0,0)	(0,1)	(0,0)	(0,0)	(1,1)
s_{r-}	(0,1)	(0,1)	(0,1)	(0,1)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(1,1)

removed first. According to the construction of our truth assignment, this means that the ith literal occurring in d is true, so the clause d is also true. As these consideration applied to all clauses, the truth assignment constructed satisfies the formula.

Example: Table 9 □

4 The Remaining Cases and Elimination Orders

Those cases without tight classifications, namely 2-WEAK, 2-Z-WEAK, 2-Z-DOMINANCE and 2-Z-SIMULTANEOUS, shall be investigated in some detail now. For this, a number of elimination problems on graphs are introduced and shown to be equivalent. By translating these problems from a game-theoretic framework into graph problems, they should be easier accessible for future work.

Definition 17 Let (G, B, E) be a bipartite graph with vertices $G \cup B$ and edges $E \subseteq G \times B$. By $N(u)$ we denote the set of neighbors of u. We let $v \in G$ be (weakly) eliminable by $u \in G$, iff $N(v) \subset N(u)$ ($N(v) \subseteq N(u)$), and $v \in B$ be (weakly) eliminable by $u \in B$, iff $N(v) \supset N(u)$ ($N(v) \supseteq N(u)$). A vertex is (weakly) eliminable, iff it is (weakly) eliminable by some vertex. A vertex is eventually (weakly) eliminable, iff there is an elimination order on the graph resulting in its elimination. Let SUBSUP (WEAKSUBSUP) denote the problem of deciding whether a given vertex is eventually (weakly) eliminable in a given bipartite graph. Finally, let LAZYSUBSUP denote the problem of deciding whether a given vertex is eventually eliminable in a given bipartite graph with an elimination order that only switches between eliminating B-vertices and G-vertices if no alternatives are left.

Proposition 1 SUBSUP *is log-space equivalent to* 2-Z-DOMINANCE, WEAK-SUBSUP *is log-space equivalent to* 2-Z-WEAK *and* LAZYSUBSUP *is log-space equivalent to* 2-Z-SIMULTANEOUS.

Proof To move from games to graphs, let G contain a vertex for each strategy of the first player, and B a vertex for each strategy of the second player. There is an edge between $(u, v) \in G \times B$ iff the first player receives payoff 1 when playing u against v. For the other direction, simply invert this construction. It is straightforward to verify that each pair of elimination concepts is equivalent. □

Definition 18 Consider a directed graph (G, E). We call $v \in G$ eliminable by $u \in G$, iff $O(v) \subseteq O(u)$ – here $u' \in O(u)$, iff $(u, u') \in E$. A vertex is eventually eliminable iff there is an elimination order on the graph resulting in its elimination. Let DIRSUB denote the problem of deciding whether a given vertex is eventually eliminable in a given directed graph. Let BIPDIRSUB be the restriction of DIRSUB to bipartite graphs.

Proposition 2 2-WEAK, BIPDIRSUB *and* DIRSUB *are log space equivalent.*

Proof To move from 2-WEAK to DIRSUB, let there be a vertex for any strategy by either player. Let there be an edge from u to v, iff u and v belong to different players, and the owner of u obtains the preferred payoff when playing u against v.

For the reduction from DIRSUB to BIPDIRSUB, split any vertex u into two vertices u_1, u_2. Replace any edge (v, u) by (v, u_1), and any edge (u, v) by (u_2, v). Add an edge (u_1, u_2).

Finally, for the direction BIPDIRSUB to 2-WEAK, assume the graph admits the partition $G = G_1 \cup G_2$. Let the first player have a strategy for each vertex in G_1, and the second player a strategy for each vertex in G_2. A player receives the higher payoff for playing strategy u against strategy v, iff there is a directed edge from u to v.

It is straightforward to verify that these reductions are actually correct. □

Corollary 1 BIPDIRSUB *and* DIRSUB *are* P*-hard.*

5 Conclusions

The classification of the computational hardness of the various forms of strategy elimination can contribute to the discussion (see e.g. [23]) of their merit as an approach to a normative theory of the behaviour of rational agents. The NP-completeness and membership in P results mirror the observations on uniqueness: The potential outcomes of iterated elimination of (weakly) dominated strategies in games with at least two (three) strategies is inherently non-unique (assuming P $\neq$ NP) – there cannot be a simple modification to regain uniqueness. On the other hand, restriction to zero-sum games does allow significant control over potential outcomes of iterated elimination.

The P-hardness results also tell us something meaningful about the respective elimination concepts: The sequentiality (one round of eliminations after the other) involved in the definition of iterated strategy elimination is essential (again under standard assumptions from complexity theory), there cannot be a simple equivalent definition without it. Thus, Theorems 3, 4 may be the most surprising results in this

paper: For games with only 2 distinct payoffs, or zerosum games with only 3 distinct payoffs, iterated elimination of strictly dominated strategies is neither trivial, nor fully sequential.

Proposition 2 now means that the alternation of elimination rounds between the two players is not an essential part of iterated elimination of weakly dominated strategies for games with just two distinct payoffs.

There is another, unique contribution the investigation of computability and complexity issues can make to game theory. In the spirit of *bounded rationality* (see e.g. [22]), one should note that in order to comply with some theoretical concept of decision making, an agent has to be able to compute its prescription (this is expanded upon in [19]). For example, finding a Nash equilibrium in a bimatrix game with real payoffs is not computable [20], hence, seems inadequate as a solution concept. Even if one only considers integer payoffs, and is satisfied with approximate Nash equilibria[11], one would still require solving a PPAD-complete problem [6, 8]. As it is believed that FP $\subsetneq$ PPAD $\subsetneq$ FNP (see [2, 18]), this would demand significant computational resources, and may be rejected as unreasonable.

Returning to the situation for iterated strategy elimination, we have to overcome the distinction between search problems and decision problems. If an agent is supposed to follow an iterated elimination rule, he needs to ascertain whether or not a given strategy is to be neglected. The NP-completeness of WEAK and DOMINANCE suggests that such a decision necessitates significant computational resources. The assumption iterated elimination of (weakly) dominated strategies were actually used by real-life agents in complicated situations subsequently appears to be unrealistic.

We can go a (very speculative) step further: According to common belief, the P-complete problems are inherently sequential, i.e. do not benefit from parallel processing. Considering that the human brain can be likened to a slow, but massively parallel computer ([5, Pages 114–115]), or by simply noting human lazyness, P-complete problems – such as STRICT – also may be unlikely to describe actual human thinking. Such reasoning would only leave iterated elimination of never best responses as a good candidate for *bounded rationality*.

Acknowledgements I am grateful to Anuj Dawar for his support and advice as my PhD advisor. Furthermore, I would like to thank the anonymous referees of an earlier version for their detailed and helpful comments.

References

1. Apt, K.R.: Order independence and rationalizability. In: Theoretical Aspects of Rationality and Knowledge (TARK X) (2005). arXiv:0509063
2. Beame, P., Cook, S., Edmonds, J., Impagliazzo, R., Pitassi, T.: The relative complexity of NP search problems. J. Comput. Syst. Sci. **57**, 3–19 (1998)
3. Bernheim, B.D.: Rationalizable strategic behavior. Econometrica **52**, 1007–1028 (1984)

[11] Note that approximate Nash equilibria are not necessarily approximations to Nash equilibria, but a far weaker criterion.

4. Brandt, F., Brill, M., Fischer, F., Harrenstein, P.: On the complexity of iterated weak dominance in constant-sum games. Theory of Computing Systems **49**(1), 162–181 (2011). doi:10.1007/s00224-010-9282-7
5. de Callataÿ, A. Natural and Artificial Intelligence, 2nd edn. North-Holland (1992)
6. Chen, X., Deng, X.: Settling the complexity of 2-player Nash-equilibrium. Tech. Rep. 134, Electronic Colloquium on Computational Complexity (2005)
7. Conitzer, V., Sandholm, T.: Complexity of (iterated) dominance. In: EC '05: Proceedings of the 6th ACM Conference on Electronic Commerce, pp. 88–97. ACM, New York (2005). doi:10.1145/1064009.1064019
8. Daskalakis, C., Goldberg, P., Papadimitriou, C.: The complexity of computing a Nash equilibrium. SIAM J. Comput. **39**(1), 195–259 (2009)
9. Downey, R., Fellows, M.: Parameterized Complexity. Springer (1999)
10. Gilboa, I., Kalai, E., Zemel, E.: The complexity of eliminating dominated strategies. Math. Oper. Res. **18**(3), 553–565 (1993). http://www.jstor.org/stable/3690089
11. Gilli, M.: Iterated admissibility as solution concept in game theory. Department of Economics Working Paper 47, University of Milan-Bicocca (2002)
12. Kalai, E., Zemel, E.: On the order of eliminating dominated strategies. Discussion Papers 789, Northwestern University, Center for Mathematical Studies in Economics and Management Science (1988)
13. Knuth, D., Papadimitriou, C., Tsitsiklis, J.: A note on strategy elimination in bimatrix games. Oper. Res. Lett. **7**(3), 103–107 (1988)
14. Marx, L.M., Swinkels, J.M.: Order independence for iterated weak dominance. Games and Economic Behavior **18**(2), 219–245 (1997)
15. Marx, L.M., Swinkels, J.M.: Order independence for iterated weak dominance. Games and Economic Behavior **31**(2), 324–329 (2000). Corrigendum
16. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT Press (1994)
17. Papadimitriou, C.H.: Computational Complexity. Addison Wesley (1994)
18. Papadimitriou, C.H.: The complexity of finding Nash equilibria. In: Nisan, N., Roughgarden, T., Tardos, É., Vazirani, V. (eds.) Algorithmic Game Theory, pp. 29–52, Cambridge University Press (2007)
19. Pauly, A.: Computable Metamathematics and its Application to Game Theory. PhD Thesis, University of Cambridge (2012)
20. Pauly, A.: How incomputable is finding Nash equilibria? Journal of Universal Computer Science **16**(18), 2686–2710 (2010). doi:10.3217/jucs-016-18-2686
21. Pearce, D.: Rationalizable strategic behavior and the problem of perfection. Econometrica **52**, 1029–1050 (1984)
22. Rubinstein, A.: Modeling Bounded Rationality. Zeuthen Lecture Book. Massachusetts Institut of Technology (1998)
23. Trost, M.: An epistemic rationale for order-independence. In: 10th Conference on Logic and the Foundations of Game and Decision Theory (2012)