A Protocol for Serializing Unique Strategies

Marcel Crasmaru[1], Christian Glaßer $^{\star 2}$, Kenneth W. Regan $^{\star\star 3}$, and Samik Sengupta[3]

[1] Tokyo Inst. of Technology, `marcelis.titech.ac.jp`
[2] Universität Würzburg, `glasserinformatik.uni-wuerzburg.de`
[3] University at Buffalo, {`regan,samik`}`cse.buffalo.edu`

Abstract. We devise an efficient protocol by which a series of two-person games G_i with unique winning strategies can be combined into a single game G with unique winning strategy, even when the result of G is a non-monotone function of the results of the G_i that is unknown to the players. In computational complexity terms, we show that the class UAP of Niedermeier and Rossmanith [NR98] of languages accepted by unambiguous polynomial-time alternating TMs is self-low, i.e., $\mathrm{UAP}^{\mathrm{UAP}} = \mathrm{UAP}$. It follows that UAP contains the Graph Isomorphism problem, nominally improving the problem's classification into SPP by Arvind and Kurur [AK02] since UAP is a subclass of SPP [NR98]. We give some other applications, oracle separations, and results on problems related to unique-alternation formulas.

1 Introduction

Consider the simple form of Nim where players alternate removing 1 or 2 stones from an initial pile of N stones, and the player who takes the last stone wins. A position N of this game is a win for the player to move iff N is not a multiple of 3, and the unique winning strategy for this player is to take the number that leaves a multiple of 3. Moreover, if the first player errs, the second player has a unique winning rejoinder. This extends to say that in the entire game tree of *all* possible plays from any initial position N, every non-terminal position reached either loses for the player to move or has a unique winning move. We term a game with this property *globally unique*, insofar as winning strategies wherever they exist are unique.

A two-person game is definable formally via an *alternating Turing machine* (ATM) M, which is coded like a nondeterministic TM but has separate *existential* and *universal* states. By convention the $\exists$-player moves first—i.e., the start state is existential—and the $\exists$-player wins a play from a starting game position x iff the computation path given by the players' moves ends in an accepting final configuration of M. Nim is coded by an ATM M that on input 0^N uses $\exists$-states and $\forall$-states that give options to mark off 1 or 2 cells, halts on reading the blank

* Supported by a postdoctoral grant from the German Academic Exchange Service (Deutscher Akademischer Austauschdienst—DAAD).

** Supported in part by NSF grants INT 9726724 and CCR-9821040.

at the end of the input, and accepts iff it just came from an $\exists$-state. The definition of x being a first-player win, i.e. being in the language $L(M)$, recursively defines an existential configuration to be positive iff it has at least one move to a positive configuration, and a universal configuration to be positive iff all moves lead to a positive configuration. Then $x \in L(M)$ iff the start configuration on input x is positive. Niedermeier and Rossmanith [NR98] defined:

Definition 1. *An ATM is* unambiguous *if every positive non-final existential configuration has exactly one move to a positive configuration, and every negative non-final universal configuration has exactly one move to a negative configuration.*

They denoted by UAP the class of languages accepted by unambiguous polynomial-time bounded ATMs, since it is the natural alternating analogue of the class UP of languages accepted by unambiguous NTMs. They showed that FewP $\subseteq$ UAP $\subseteq$ SPP.[1] The polynomial time bound on the ATMs enforces that each play of the corresponding game finishes within a reasonable amount of time. Thus languages in UAP comprise all denotations of reasonable globally-unique games.

The main intellectual problem solved by this paper is whether highly general combinations of globally-unique games are globally unique. Suppose Player 1 and Player 2 play a tournament G^* of Nim-style games $G_1, \ldots, G_m$, under the strange condition that the winner of the tournament is determined by an unknown Boolean function $f(x_1, \ldots, x_m)$ of the results of the individual games. The function f from Player 1's perspective may even be non-monotone, meaning that it could benefit Player 1 to play to *lose* the first game in order to win the tournament. Note that in Definition 1 there is no constraint on existential configurations (Player 1 to move) that are negative, nor on universal configurations (Player 2 to move) that are positive. What general rule and incentive can one give to induce uniquely optimal play in a position of G_1 where the player to move is losing?

Difficulties can be seen in the case of just two games where f is OR, so that Player 1 wins G^* if he wins either G_1 or G_2. The initial position of G^* may be a pair of initial positions σ_1, σ_2 of G_1 and G_2 respectively, except that σ_2 may not be known to the players at the start of G_1, and may even depend on the *outcome* of the actual play of G_1. If σ_1 and σ_2 are (both known and are) first-player wins, Player 1 seems to have two winning strategies: win G_1, or not care about G_1 and win G_2. We can attempt to force unique play in G by requiring Player 1 to win the winnable game with lowest index. But now what happens if Player 1 makes a mistaken move in some position π_1 of G_1 (possibly $\pi_1 = \sigma_1$ itself), and the given σ_2 turns out to be a second-player win? Now Player 2

[1] FewP is the class of languages accepted by polynomial-time NTMs that have only polynomially-many accepting computations on any input. SPP [FFK94] comprises those L defined by polynomial-time NTMs whose number of accepting computations either equals or surpasses by 1 its number of rejecting computations, with the latter case putting an input x into L.

seems to have two winning strategies: point out Player 1's mistake in G_1 or wait for G_2. Because G_1 is a globally-unique game, Player 2 *can* prove the mistake by switching roles with Player 1 and showing that some other move wins from π_1. We can stipulate that Player 2 must take the "prove a mistake" option when it works. But what if Player 2 errs by not doing so? Since any error must be punished uniquely, we must allow Player 1 to be able to win by pointing out his own mistake in G_1 at π_1. In case σ_2 turns out to be a first-player win after all, we must similarly stipulate that Player 1 must take the point-out-mistake option. But what if Player 1 now errs by not doing so? Can we avoid an infinite regress of artificial new rules?

When the number m of constituent games varies with the complexity parameter "n," there is also the danger that permitting players to review previous play will lead to exponential backtracking. The protocol for converting general alternating-P (i.e., PSPACE) computations into plays of globally-unique games in [ACRW02] involves exponential backtracking—it cannot be improved to polynomial unless PSPACE collapses into UAP, hence into SPP, thence into $\oplus$P and PP, which is generally disbelieved. The only advantage in our setting is that the individual games G_i already have global uniqueness of winning strategies.

Main Theorem 1. Given unambiguous polynomial-time ATMs defining games $G_1, \ldots, G_m$, and a polynomial-time referee function f, we can construct an unambiguous polynomial-time ATM M^* such that for all starting positions $x_1, \ldots, x_m$ of $G_1, \ldots, G_m$ respectively, $(x_1, \ldots, x_m) \in L(M^*) \iff (x_1, \ldots, x_m)$ wins for the $\exists$-player in G^*. Moreover, only a final deterministic part of M^* depends on f.

We prove this in greater generality by allowing $x_1, \ldots, x_m$ to arise dynamically from an overall starting position x given to M^*. Technically M^* becomes an *oracle ATM* and $x_1, \ldots, x_m$ become the *oracle queries* made by M^*—and the difference is that the oracle computation can be *adaptive*. "Adaptive" means that the starting position x_i for G_i can depend on the play of lower-indexed games, and so not be a fixed function of the input x. We lose no generality by supposing that "i" is encoded into x_i and all subsequent positions of each game G_i, so that we can picture a single globally-unique process controlling the constituent games.

Doing so lends our main theorem the denotation that the class UAP is *self-low*, and we state and prove it this way in the next section. Our proof still relies on the game formulation embodied in the statement of Theorem 1. A standard strategy for self-lowness in complexity theory is to show the class to be closed under (complements and) polynomial-size conjunctions and under "'one query," but in the adaptive case this strategy makes a step of existentially guessing queries that we do not know UAP a-priori to be closed under (note that NP $\subseteq$ SPP is disbelieved in [FFK94]). Moreover we have not found a great simplification of our proof even just for conjunctions as posed above—and closure of UAP under intersection was not noted in [NR98].

2 Main Theorem

For any oracle set A, UAP^A is the class of languages accepted by polynomial-time oracle ATMs that are unambiguous with oracle A. We re-state:

Theorem 1. $\mathrm{UAP}^{\mathrm{UAP}} = \mathrm{UAP}$.

Proof. Let M be a polynomial-time bounded oracle ATM that has the global uniqueness property with oracle $A \in \mathrm{UAP}$, and let M_A be a polynomial-time bounded non-oracle ATM with global uniqueness that accepts A. The following protocol implicitly defines a polynomial-time bounded non-oracle ATM M^* with global uniqueness that accepts $L(M^A)$. The $\exists$-player and $\forall$-player initially and always make their corresponding moves in M until M makes a query. The sub-game G_i arises when M makes the i-th query y_i in its computation path, and commences from the position σ_i corresponding to the initial configuration of M_A on input y_i. Although the games G_i may appear uniform and dependent since M_A is the same machine in all instances, the query strings y_i may contain the actual rules of the game G_i with M_A used intuitively only as a universal reader. Thus no generality seems to be lost by our technical setting, compared to the motivational description above.

Before describing the protocol, we need to define its ingredients. First, a global total ordering of legal moves at every possible position of every game G_i is imposed. We picture the lowest legal move at a position as "leftmost." The $\exists$-player and $\forall$-player play each G_i in sequence, except that either has the option at any time to "trump previous play." At every move of a game G_i, the player to move *declares* "Win" or "Lose"—and in the latter case, *must* play the leftmost legal move on pain of otherwise being liable for "trumping." If the plays of each G_i and the intervening simulation of M all finish without either player using the trump option, then the overall winner is determined by the final configuration of M that is reached. When M is an oracle P-machine, it helps to picture M as a neutral referee who determines the overall winner based only on the outcomes of the games (i.e., queries).

Trumping means to backtrack to a previous position π in some game G_j—with either the same or the opposite player to move at π—and change the declaration and/or move that was made there. The essence is that the following three rules avoid the infinite-regress problems described in Section 1.

(a) If the original player to move at π declared "Lose," then either player can trump by declaring "Win" and playing a move.
(b) If the original player to move at π declared "Win," then either player can trump by declaring "Win" and playing a different move.
(c) If the original player to move at π declared "Win," then either player can trump by playing that move, declaring "Win" at the succeeding position π', and making a move from π'.

Once "trump" is declared at π, future "trump" options may return only to games G_i with $i < j$, to positions earlier than π in the original play of G_j, or:

In any ensuing play after a case-(c) trump move, it is legal to make a "counter-trump" move of type (b) by returning to π, declaring "Win," and making a move—which again must be different from the move that was originally made at π.

In all cases, if the trumping player wins the ensuing play of game G_j, then he wins the entire game G^*. Note, however, that terminal positions of G_j still allow the above (counter-)trumping options. The key idea of the proof is that *whenever a victorious trumping move is legal, it is the unique optimal play.*

These provisos ensure that plays of the expanded game G^* have length polynomial in both m and the maximum length of any play of any G_j. Every position of the game G^* encodes not only the corresponding position of the current game G_j, but also the entire game history of G^* to that point—this in particular determines all possible trumping moves. Moreover, the number of legal (trumping) moves available in any position expands by only a polynomial factor. This completes the description of the game G^*.

To prove that G^* has the global uniqueness property—regardless of the referee's verdict in plays where all m sub-games are completed without trumping—we make the following observations:

1. There is a unique play α_0 of the sequence of games that allows no successful trumping opportunities. In this play, each player makes the correct declaration at every step, and a player declaring "Win" makes the unique winning move. The play α_0 has no trumping moves itself.
2. If a play α deviates from α_0 at a position π by an incorrect move or declaration, then in any play forward from π, either player can win by rewinding to position π and trumping. Here are the possible deviations and trump responses:
   (a) Declaring "Lose" in a winning position. Trump by declaring "Win" and playing a winning move. Trump option (a) is the only one that applies, and by global uniqueness of G_j itself, the winning move is unique. (It may be the leftmost move itself—i.e., the one that was played.)
   (b) Declaring "Win" in a winning position, but not playing the winning move. Trump by declaring "Win" and playing the correct move—which again is unique by global uniqueness of G_j. This option (b) is the unique correct reply.
   (c) Declaring "Win" in a losing position. Trump by playing the move that was played (which need not be the leftmost move), then declaring "Win," and making a winning move. Trump option (a) doesn't apply and (b) is false, so (c) is unique—as is the winning move in the second position.

The other kind of deviation from α_0 that can occur is a trumping move. We distinguish between a "virgin" deviation and one in reply to an earlier deviation at the same position π. We first argue inductively that a virgin deviation (by trumping) always loses for the trumping player. If it is a trumping move of type (a) or (b), then the resulting position is a win for the responding player. Since (by induction) trumping at a position earlier than π along α_0 would be virgin and thus lose, and since no "counter-trump" at π is available, winning the ensuing play of the game G_j is the only winning option for the responder—and it is unique by global uniqueness of G_j. If the virgin deviation is of type (c), then the position π is winning in G_j for the player P to move there, and P originally made the right move μ—because it was not a deviation. The trumper makes μ and then declares that the resulting position π' is a win for him, making a move μ' there. Here the trumper is wrong about π', so the responder can win the ensuing play of G_j. The responder does have the "counter-trump" option available, but since μ *was* the right move, it would lose. Thus winning the ensuing play of G_j is the unique way to punish a virgin deviation of type (c).

To finish the argument, consider any position ρ in which the player P to move can win. If ρ is along the unique optimal play α_0, then all alternatives to furthering α_0 lose, so *a fortiori* the unique winning move must be the one that furthers α_0. If not, then there is a first position π at which the play α deviates from α_0. If the deviation at π was by a bad move, or by a move accompanying the wrong declaration, and there has been no trumping at π already, then trumping at π is legal and winning for either player. Since all other moves by P either leave trumping at π open to the other player, or are virgin trumps at positions earlier than π along α_0, trumping at π is the unique winning move for P.

Otherwise, a trumping move has occurred at π, and position ρ is immediately afterward or in ensuing play from that move. If the trump move was of type (a) or (b), virgin or not, then since there is no counter-trump at π, and since trumping below π is virgin and hence losing, there is no option other than the ensuing play in game G_j, whose global uniqueness carries through here. If it was a virgin trump of type (c), then it was incorrect, and ensuing play is globally unique as argued above. It remains to cover cases where ρ ensues from a trump of type (c) at π that comes after an incorrect "Win" declaration and/or move at π that represented the original deviation from α_0.

(i) The "Win" declaration was incorrect—i.e., position π is losing in G_j for the player to move. In this case the type-(b) counter-trump option loses. As in the cases above of a type (a) or (b) trump at π, the only way ρ can be winning for the player to move is for it to be winning in G_j, and the unique winning play in G_j is the unique way to win G^*.

(ii) The "Win" declaration was correct, but the original player made the wrong move. Then the counter-trump option at π is winning. Since it is available to both players at all times in the ensuing play of G_j, it is always the unique optimal reply.

This finishes the argument that G^* represents a general protocol for playing a series of globally-unique games in a globally-unique way. Note the symmetry between the two players in all aspects of the argument, and the use of "leftwardness" *only* to constrain options when a player declares "Lose." This already amounts to a proof that $\mathrm{P}^{\mathrm{UAP}} = \mathrm{UAP}$.

For $\mathrm{UAP}^{\mathrm{UAP}} = \mathrm{UAP}$ we need only describe the machine M^* a little further. If a trumping move occurs during the play, then M^* follows the resolution of this trump—and all possible counter-trumps and trumps at earlier positions—and gives the result as its own final answer (accept iff the $\exists$-player wins). If not, then the play β_i from σ_i ends with a winner of that game, and M^* takes the result as the answer to the oracle query y_i. If the sub-games for all queries reach their conclusion and no trumps are made, then eventually a terminal position of the game G_M represented by M is reached. If it is winning for the player who just moved, then the opposing player still has the option of trumping; if not, then the game ends and the opposing player wins the game G_{M^*} right there. (This last proviso preserves global uniqueness even at the terminal position of G_{M^*}.)

For each oracle query y_i, there is a unique "correct" play α_i from σ_i, being a segment of "α_0" as established above. If β_i deviates from α_i, then in any ensuing configuration of M^*, either side has the winning option of trumping at the (first) point of deviation, and this winning option is unique. If not, then M^* gets the correct answer to the oracle query $y_i \in ?A$. Putting this all together, the global uniqueness that M has with oracle A carries through to global uniqueness of M^*, and $L(M^*) = L(M^A)$. $\square$

We remark that coming into or exiting a sub-game G_i may give two consecutive moves to one of the players, but (i) this does not matter to the above analysis, (ii) it is easy to avoid beforehand by modifying M_A and M to avoid this, and (iii) it is fixable afterwards by padding methods used here and in results of [ACRW02] quoted below anyway.

Corollary 1. UAP *is closed under all Boolean operations.*

One consequence of self-lowness is that several hierarchies based on Turing reductions with uniqueness are contained in UAP. Niedermeier and Rossmanith [NR98] define "$AU\Sigma_k^p$" to be the subclass of UAP of languages defined by games that start with the $\exists$-player and have at most $k-1$ alternations in any play between the $\exists$-player and the $\forall$-player. They credit to "Hemaspaandra [unpublished]" the observation that these coincide with the levels $\mathrm{UP}, \mathrm{UP}[\text{co-UP}], \mathrm{UP}[\text{co-UP}[\mathrm{UP}]], \ldots$ defined by quantifiers on UP and co-UP predicates. Lange and Rossmanith [LR94] define "$AUPH$" to be the union of these classes, "UPH" to be the union of the levels $\mathrm{UP}, \mathrm{UP}^{\mathrm{UP}}, \mathrm{UP}^{\mathrm{UP}^{\mathrm{UP}}}, \ldots$, and "$\mathcal{UPH}$" to be the union of the "smart-reduction" levels $\mathrm{UP}, \mathrm{UP}_s^{\mathrm{USAT}}, \mathrm{UP}_s^{\mathrm{USAT}_2}$ (where "USAT_2" characterizing $\mathrm{UP}_s^{\mathrm{USAT}}$ was described above following Theorem 3). They give SPP as an upper bound for these hierarchies, and Theorem 1 allows us immediately to improve this to:

Corollary 2. $\mathcal{UPH} \subseteq \mathrm{UAP}$*, and also* $\mathrm{UAP}_s^{\mathrm{USAT}} = \mathrm{UAP}$.

Our main corollary, with application to the complexity of the Graph Isomorphism problem, needs its own section.

3 Logic games, smart reductions, and graph isomorphism

In [ACRW02], UAP was characterized as the class of languages that polynomial-time many-one reduce to the following promise problem *GUQBF*:

INSTANCE: A Boolean formula F in variables $x_1, \dots, x_d$ for some $d \geq 0$ that induces the quantified Boolean formula $\psi = (\exists x_d)(\forall x_{d-1}) \cdots (Qx_1)\ F$.
PROMISE: The logic game on ψ has the global uniqueness property.
QUESTION: Is ψ true?

Here "Q" is $\exists$ if d is odd, else it is $\forall$. In the usual logic game, the $\exists$-player goes first and assigns 0 or 1 to x_d, then the $\forall$-player assigns to x_{d-1}, and the players alternate moves until F is either made true, a win for $\exists$, or false, a win for $\forall$. A family of small formulas (wins for $\exists$) in which the promise holds is typified by

$$F = x_1 \ \vee \ x_2(x_3 \ \vee \ x_4(x_5 \ \vee \ x_6(x_7 \ \vee \ x_8))),$$

where we have written AND as multiplication binding tighter than OR to improve visual intuition. The $\exists$-player must assign 1 to even-numbered variables, whereupon the $\forall$-player's next choice is immaterial; but if the $\exists$-player wrongly assigns 0 to (say) x_8, then the $\forall$-player succeeds uniquely by assigning 0 to x_7 and so on, whereupon the $\exists$-player's moves become immaterial and losing.

The promise problem USAT can be regarded as a sub-promise of *GUQBF*:

INSTANCE: A Boolean formula F in variables $x_1, \dots, x_d$, here viewed as inducing the quantified Boolean formula $\psi = (\exists x_d)(\exists x_{d-1}) \cdots (\exists x_1) F$.
PROMISE: F has zero or one satisfying assignment (which implies that the logic game on ψ has the global uniqueness property).
QUESTION: Is F satisfiable—i.e., is ψ true?

Abstractly, a promise problem (Q, R) has *promise set* Q and *property set* R, and a language S is a *solution* if $S \cap Q \subseteq R$ and $Q \setminus S \subseteq \bar{R}$. A reduction to a promise problem is required to be simultaneously a reduction to every solution. For a many-one reduction f this entails $Ran(f) \subseteq Q$ [Sel88], but for reductions of Turing type it is possible that allowing queries to strings outside of Q makes a difference, as discussed by Grollman and Selman [GS88]. They called a Turing reduction to a promise problem *smart* if it never queries strings outside the promise set. A many-one reduction is a special case of a *smart* Turing reduction.

Note that a language A belongs to UP if and only if $A \leq_m^p$ USAT, so that USAT characterizes UP by many-one reductions the same way *GUQBF* characterizes UAP. However, the class $\mathrm{P}_s^{\mathrm{USAT}}$ of languages with smart polynomial-time Turing reductions to USAT is apparently larger: it is closed under complements, contains FewP [CHV93],[2] and contains the graph-isomorphism problem:

[2] Here and in [NR98] the class is written "$\mathrm{P}^{\mathcal{UP}}$," but we write "$\mathrm{P}_s^{\mathrm{USAT}}$" to avoid possible font confusion with P^{UP} and to emphasize the role of USAT.

Theorem 2 (after [AK02]). $\mathrm{GI} \in \mathrm{P}_s^{\mathrm{USAT}}$.

Proof. The main lemma of [AK02] creates an oracle reduction to a group-theoretically defined language L such that every query w to L has a unique witnessing answer. Arvind and Kurur remark that the queries are "UP-like." Because their language L belongs to NP, the w can be transformed to queries w' to USAT, and the "UP-like" property is preserved and makes this a "smart" polynomial-time Turing reduction to USAT. Their algorithm computes what they call a "UP-single-valued function with SPP oracle," but the use of this function to decide Graph Isomorphism stays within the bounds of $\mathrm{P}_s^{\mathrm{USAT}}$. □

Corollary 3 (to Theorem 1). $\mathrm{P}_s^{\mathrm{USAT}} \subseteq \mathrm{UAP}$ *(hence* $\mathrm{GI} \in \mathrm{UAP}$*).*

Proof. The oracle P-machine accomplishing a smart reduction to USAT initiates only those plays of the logic game on existential formulas that have (global) uniqueness, so it becomes a non-oracle UAP-machine via Theorem 1. □

4 Globally Unique Formulas

We note first that the promise problem *GUQBF* is invariant under equivalence of Boolean formulas F and F', since the PQBFs induced from F and F' define the same logic game. Thus we can characterize game positions with global uniqueness by selecting representatives from each equivalence class. Let TRUE (resp., FALSE) denote a constant Boolean formula whose value is 1 (resp., 0). We define inductively $A_0 = \{ \text{FALSE} \}$, $B_0 = \{ \text{TRUE} \}$, and for $d \geq 1$,

$$\begin{aligned} A_d &= \{ (x_d \ \wedge \ \neg F_1) \ \vee \ (\bar{x}_d \ \wedge \ \neg F_0) : F_0, F_1 \in B_{d-1} \}, \\ B_d &= \{ (x_d \ \wedge \ \neg F_1) \ \vee \ (\bar{x}_d \ \wedge \ \neg F_0) : \\ &\qquad (F_0 \in A_{d-1} \text{ and } F_1 \in B_{d-1}) \text{ or } (F_0 \in B_{d-1} \text{ and } F_1 \in A_{d-1}) \}. \end{aligned}$$

Here B_d comprises those d-variable Boolean formulas that induce globally-unique logic games with the ∃-player to move that are wins for the ∃-player, while A_d comprises those in which the ∃-player is to move but loses. Note that A_d says that both substitutions for x_d leave Boolean formulas whose *negations* are in B_{d-1}, meaning the negations are unique wins with the ∃-player to move, which implies that the resulting formulas themselves are unique ∀-player wins with the ∀-player to move. The recursion for B_d is interpreted similarly.

For example, $A_1 = \{ (x_1 \ \wedge \ \text{FALSE}) \ \vee \ (\bar{x}_1 \ \wedge \ \text{FALSE}) \}$, which is equivalent to $A_1 = \{ \text{FALSE} \}$ and (mentioning x_1) to $A_1 = \{ x_1 \ \wedge \ \bar{x}_1 \}$. Also

$$B_1 = \{ (x_1 \ \wedge \ \text{TRUE}) \ \vee \ (\bar{x}_1 \ \wedge \ \text{FALSE}), \quad (x_1 \ \wedge \ \text{FALSE}) \ \vee \ (\bar{x}_1 \ \wedge \ \text{TRUE}) \},$$

which reduces to $B_1 = \{ x_1, \bar{x}_1 \}$. The following is shown in [ACRW02] by straightforward induction.

Lemma 1 ([ACRW02]). *The formulas in $A_d \cup B_d$ are pairwise inequivalent, and when interpreted as logic games with the $\exists$-player to move, those in A_d are $\exists$-player losses with global uniqueness, while those in B_d are $\exists$-player wins with global uniqueness.*

Every formula in A_d has exactly N_d satisfying assignments, where $N_d = (2/3)(2^d - 1)$ if d is even, while $N_d = N_{d-1}$ if d is odd. Moreover, every formula in B_d has exactly $N_d + 1$ satisfying assignments. The journal version [ACRW02] shows that UAP $\subseteq$ SPP also follows from this. Counting A_d and B_d shows they contain Boolean functions all of whose formulas have bit-size $\Omega(2^d)$.

Here we study the analogues of A_d and B_d for polynomials in arithmetic mod 2, under the standard correspondence $H(\textsc{True}) = 1$, $H(\textsc{False}) = 0$, $H(x_i) = x_i$, $H(\neg f) = 1 - H(f)$, $H(f \wedge g) = H(f)H(g)$, and $H(f \vee g) = H(f) + H(g) - H(f)H(g)$. Define I_d to be the ideal generated by $\{ x_1^2 - x_1, \ldots, x_d^2 - x_d \}$ in $\mathcal{F}[x_1, \ldots, x_d]$, where we take the field $\mathcal{F}$ to be the integers mod 2 or any field of characteristic 2. We observe (proofs from here on are in the full paper):

Lemma 2. *Over $\mathcal{F}$, for $d \geq 1$, every polynomial in $H(A_d)$ has degree $d-1$ modulo $\mathcal{I}_d$, while every polynomial in $H(B_d)$ has degree d modulo $\mathcal{I}_d$.*

It is curious that the cancellation is guaranteed only in characteristic 2. This leads to the following decision problem "Degree Mod $\mathcal{I}_n$" about representations of Boolean functions by polynomials, which we have not seen studied in the literature on the "polynomial method" (see [Bei93] for stem references).

Instance: A formula for a polynomial $f \in \mathcal{F}[x_1, \ldots, x_n]$.
Question: Does f have degree n when reduced modulo the ideal $\mathcal{I}_n$?

Theorem 3. Degree Mod $\mathcal{I}_n$ *is polynomial-time many-one hard for* UAP.

What is the exact complexity of Degree Mod $\mathcal{I}_n$? Note that f is a tautology or unsatisfiable iff $H(f)$ reduces to 1 or 0, so that the related question of whether the degree mod $\mathcal{I}_n$ is *positive* is NP-hard. It seems not to follow simply, however, that our problem of whether the degree is n is NP-hard. The problem of whether a given set of polynomials reduces to 1 under the Gröbner basis algorithm belongs to the second level of the polynomial hierarchy [Koi96], but the same for Degree Mod $\mathcal{I}_n$ would put UAP inside PH, which also seems questionable. Nor do we even know whether Degree Mod $\mathcal{I}_n$ belongs to polynomial space! This problem deserves further study, and its hardness for UAP provides a context.

Now define $\mathcal{A}_d$ to be the set of d-variable Boolean formulas that are equivalent to a formula in A_d, and $\mathcal{B}_d$ similarly with regard to B_d. Then set $\mathcal{A} = \cup_d \mathcal{A}_d$, $\mathcal{B} = \cup_d \mathcal{B}_d$. Clearly both languages are UAP-hard, and by the recursive definitions, $\mathcal{A} \cup \mathcal{B}$ is polynomial-time self-reducible. We prove something stronger:

Theorem 4. *$\mathcal{A}$ and $\mathcal{B}$ are polynomial-time isomorphic and self-reducible.*

It is not clear whether these equations yield conjunctive or disjunctive self-reductions that are polynomially well-founded. Many other questions pop to mind about the sets $\mathcal{A}$ and $\mathcal{B}$. Are they NP-hard? Is their complexity tied to that of the promise problem *GUQBF*? Are they learnable? We note:

Proposition 1. *For every formula $F \in \mathcal{B}_d$ with d even, there are $2^{d/2}$ satisfying truth assignments α such that flipping the value on that assignment to* FALSE *leaves a formula in $\mathcal{A}_d$.*

The $2^{d/2}$ assignments are those reached by unique winning play by $\exists$ against the $2^{d/2}$ possible different plays by the $\forall$-player. Being able to compute any one of them deterministically is equivalent to telling whether the QBF induced from F is true, i.e. to solving *GUQBF* on that instance. Thus unless UAP = P these ranges must avoid polynomial-time computable functions in some sense. Does that give them any pseudo-random properties?

5 Oracles and the UAP vs. SPP problem

Since SPP is intuitively just above UAP and is also self-low, the natural next question to attack is whether UAP = SPP. Niedermeier and Rossmanith [NR98] noted that UAP meets their definition of *locally definable* classes, while SPP is the smallest class thrust out by a notion of *gap-definability* [FFK94] that seemed (to them) inherently non-local. Put simply, the difference is that the gap-counting condition that defines SPP need only count the results of computation paths, while that defining UAP depends on all internal configurations of the computation tree. Thus equality may seem surprising. Inequality, however, would mean that we have found another self-low counting class, one below the level of "gap-definability" (see discussion in [FFK94,FFL96]).

We have separated UAP by oracle from its lower neighbor $\mathrm{P}_s^{\mathrm{USAT}}$. This follows on constructing an oracle A such that $\mathrm{UP}^{\mathrm{UP}^A} \not\subseteq \mathrm{P}^{\mathrm{NP}^A}$, which is not subsumed by oracle results for UP-based hierarchies in [CHV93] and [NR98]:

Theorem 5. *There exists an oracle relative to which* $\mathrm{UAP} \neq \mathrm{UP}_s^{\mathrm{USAT}}$.

This extension makes plausible that further levels of the smart-reductions hierarchy "$\mathcal{UPH}$" are also different from UAP. The proof is in the full paper.

6 Conclusions and Open Problems

Our results enhance the value of UAP as a natural complexity class. It resides just below SPP and is likewise self-low. The jump from UAP to SPP, however, may go from a large "locally-definable" class [NR98] to the smallest gap-definable class [FFK94]. If they are equal, this contrast makes the equality interesting; if they are different, why are they so structurally alike? We have tried to separate them by an oracle, and not found it easy. Even if UAP turns out to equal SPP, our protocol itself—which appears markedly different from the proof of self-lowness of SPP in [FFK94]—would likely retain independent interest.

The property of global uniqueness itself—of an ATM or of (the QBF induced by) a Boolean formula—also deserves study. Is *GUQBF* a case of a promise problem with a promise more difficult, no more difficult, or incomparable with the problem being solved? Does the promise problem $GUQBF$ even have solutions in UAP? The isomorphic languages $\mathcal{A}$ and $\mathcal{B}$ of Boolean formulas seem to have many interesting properties for further exploration.

Last, we look to further applications of the game-playing protocol behind Theorem 1. It remains serial even in the non-adaptive case where the initial game positions $x_1, \ldots, x_m$ are known in advance, since "trumping" relies on a fixed total ordering of the games G_i and temporal order of their plays. Extending it for parallel or "asynchronous" plays of the G_i might impact (unambiguous) computation in classes below P.

Acknowledgments We thank Steve Fenner for helpful conversations at Complexity'02 and afterward, and several authors of [CHV93,LR94,NR98] for answering our precedent-search queries. We thank all referees for helpful comments.

References

[ACRW02] S. Aida, M. Crasmaru, K. Regan, and O. Watanabe. Games with a uniqueness property. In *Proc. 19th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2285 of *Lect. Notes in Comp. Sci.*, pages 396–407. Springer Verlag, 2002.

[AK02] V. Arvind and P. Kurur. Graph isomorphism is in SPP. In *Proc. 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 743–750, 2002. Also ECCC TR02-037.

[Bei93] R. Beigel. The polynomial method in circuit complexity. In *Proc. 8th Annual IEEE Conference on Structure in Complexity Theory*, pages 82–95, 1993. Revised version, 1995.

[CHV93] J.-Y. Cai, L. Hemachandra, and J. Vyskoc. Promise problems and guarded access to unambiguous computation. In *Complexity Theory: Current Research, Edited by Klaus Ambos-Spies, Steven Homer, and Uwe Schöning, Cambridge University Press*. Springer Verlag, 1993.

[FFK94] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *J. Comp. Sys. Sci.*, 48:116–148, 1994.

[FFL96] S. Fenner, L. Fortnow, and L. Li. Gap-definability as a closure property. *Inform. and Comp.*, 130:1–17, 1996.

[GS88] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. *SIAM J. Comput.*, 17:309–335, 1988.

[Koi96] Pascal Koiran. Hilbert's Nullstellensatz is in the polynomial hierarchy. *Journal of Complexity*, 12(4):273–286, December 1996.

[LR94] K.-J. Lange and P. Rossmanith. Unambiguous polynomial hierarchies and exponential size. In *Proc. 9th Annual IEEE Conference on Structure in Complexity Theory*, pages 106–117, 1994.

[NR98] Rolf Niedermeier and Peter Rossmanith. Unambiguous computations and locally definable acceptance types. *Theoretical Computer Science*, 194(1–2):137–161, 1998.

[Sel88] A. Selman. Promise problems complete for complexity classes. *Inform. and Comp.*, 78:87–98, 1988.