On Domination and Control in Strategic Ability

Damian Kurpiewski, Michał Knapik, and Wojciech Jamroga
Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland
d.kurpiewski,m.knapik,w.jamroga@ipipan.waw.pl

ABSTRACT

We propose a technique to compare partial strategies for agents and coalitions in multi-agent environments with imperfect information. The proposed relation mimics the game-theoretic notion of strategic dominance: a stronger partial strategy can replace a weaker one in a full strategy, while preserving the enforceability of a given goal. Our version of dominance is based on a new notion of input/output characteristics for partial strategies. Intuitively, the inputs are the states where the strategy gains control over execution, and the outputs are the states where it returns the control to the environment. Moreover, we identify the sources of dependence between partial strategies, and show conditions under which they can be analysed in a fully independent way. We evaluate our approach on two scalable benchmarks from the literature. The experiments are carried out for the tasks of incremental synthesis of uniform strategies and optimization of a winning strategy, with very promising results.

KEYWORDS

alternating-time temporal logic, imperfect information, strategy synthesis, model checking

ACM Reference Format:
Damian Kurpiewski, Michał Knapik, and Wojciech Jamroga. 2019. On Domination and Control in Strategic Ability. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

1 INTRODUCTION

Most contemporary systems involve social as much as technological aspects, and even those that focus on technology are often based on autonomous components exhibiting self-interested, goal-directed behavior. The field of *multi-agent systems (MAS)* studies the whole spectrum of related phenomena. The theoretical foundations are traditionally based on game theory and formal logic.

As the systems around us become more complex, and at the same time more autonomous, the need for unambiguous specification and automated verification rapidly increases. Many relevant properties of multi-agent systems refer to *strategic abilities* of agents and their groups. *Logics of strategic reasoning* provide powerful tools to reason about such aspects of MAS [3, 8, 35, 39]. A typical property that can be expressed says that *the group of agents A has a collective strategy to enforce temporal property φ, no matter what the other agents in the system do*. In other words, *A* have a "winning strategy" that achieves φ on all its possible execution paths. Specifications in agent logics can be then used as input to *model checking*, which makes it possible to verify the correct behavior of a multi-agent system by an automatic tool [21, 22, 26, 31].

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

Verification and reasoning about strategic abilities is difficult for a number of reasons. The prohibitive complexity of model checking and strategy synthesis is a well known factor [13, 24, 34]. This problem can be mediated to some degree by using efficient symbolic methods and data structures [12, 17, 26, 38] that enable continuing adoption of model checking in the analysis of real-life systems. However, real-life agents typically have limited capabilities of observation, action, and reasoning. Thus, it is natural to assume that an agent can only make decisions based on her internal state and the observed part of the environment, i.e., under the conditions of imperfect information. This in turn brings its own set of difficulties. Firstly, the theoretical complexity of model checking of strategic properties in the setting of imperfect information ranges from **NP**–complete through $\Delta_2^{\mathbf{P}}$ to undecidable [27, 39], depending on the precise setup of the problem. Secondly, it is known that there is no fixed-point characterisation of typical properties of interest under imperfect information [16, 23]. Hence, most approaches to synthesis and verification boil down to checking all the possible strategies [19, 32, 37]. Unfortunately, the strategy space is huge – usually larger than the state space by orders of magnitude.

Some strategies are better than others. Clearly a strategy that "wins" on all paths is better than one that does not. In game-theoretic terms: the former *dominates* the latter. Among ones that achieve the goal, possible criteria of domination include: readability, cost-efficiency with respect to given resources, complexity of the strategy, and its robustness (how well it might handle small perturbations in the model). Here, we propose and study another criterion that refers to *tightness* of the strategy. Intuitively, those strategies are better which have a tighter control on the system dynamics.

Technically, this is defined by introducing a new concept of *input/output characteristic of a (partial) strategy*. Intuitively, the *inputs* of a strategy consist of all the states where the strategy is granted the full control over the execution of the system. To each of these entry points we assign the set of states where the strategy returns the control to the environment, i.e., the *outputs*. When treating strategies as black boxes a new strategy is better than the original one if it assigns smaller outputs to the same inputs. While the notion of dominance based on the comparison of input/output characteristics is sound, i.e., a dominating strategy can achieve at least what the dominated one can, it does not necessarily lead to simpler strategies. We thus combine the theoretical concept with heuristics geared towards simplicity of strategies.

We propose two practical algorithmic applications of the presented theory. Firstly, we design and evaluate an on-the-fly model checking algorithm that tries to synthesise a winning strategy. The main routine is based on depth-first search and synthesis, starting from the initial state. The novelty of the approach consists in elimination of dominated partial strategies that are candidates for including in the final result. This substantially reduces the search space. Secondly, we utilise a similar methodology in performing

optimization of a winning strategy, given as an input together with a model. The optimization is performed w.r.t. a threshold quality condition and heuristic preorders on partial strategies.

Related Work. To our best knowledge, both the theoretical proposal in this paper (a notion of strategy dominance based on tightness of the outcome set) and its practical application (an algorithm for dominance-based strategy synthesis for imperfect information) are entirely new. Below, we list the works that have partly overlapping concerns, albeit addressed in a completely different way.

The complexity, quality, and the number of winning strategies have been the subject of much research. Among these, cost-efficiency was considered in [1, 2, 14, 15] for strategic properties of agents with bounded resources. Permisiveness of strategies in parity games was addressed in [7, 9]. Moreover, [33] dealt with the existence of multiple winning Nash equilibrium strategies.

Readability of strategies and simplicity of their representations based on BDDs have been studied in [10, 11]. A practical framework presented in [36] allows for incremental automata-based learning of small strategies. Small (i.e., based on regular languages) strategies were also addressed in [30]. In [6], a method for synthesising solutions of Quantified Constraint Satisfaction Problems that attempts to find most preferred ones was presented.

Last but not least, several frameworks have targeted the verification of strategic properties under imperfect information. Regarding the available tools, the state-of-the-art MAS model checker MCMAS [32] combines efficient symbolic representation of state-space using Binary Decision Diagrams (BDDs) with exhaustive iteration over uniform strategies. A prototype tool SMC [37] employs bounded unfoldings of transition relation with strategy exploration and calls to MCMAS. A similar approach based on maximisation of partial strategies towards building a winning strategy is presented in [18–20]. Another prototype tool [29] avoids the brute-force strategy search by using fixed-point approximations of the input formulas. It should be noted, however, that the approximations do not always provide conclusive results. Finally, papers [4, 5] provide abstraction and bisimulation-based methods for simplifying models with incomplete knowledge.

2 PRELIMINARIES

We first recall the basic formal models of MAS, and provide a notion of partial strategy.

Models. Imperfect Information Concurrent Game Structure or *iCGS* [3, 39] is a tuple $M = \langle \mathbb{A}\mathrm{gt}, St, q_{init}, \mathcal{P}, V, Act, d, o, \{\sim_a \mid a \in \mathbb{A}\mathrm{gt}\}\rangle$ which includes a nonempty finite set of all agents $\mathbb{A}\mathrm{gt} = \{1, \ldots, k\}$, a nonempty set of states St, the initial state $q_{init} \in St$, a set of atomic propositions $\mathcal{P}$ and their valuation $V\colon \mathcal{P} \to 2^{St}$, and a nonempty finite set of (atomic) actions Act. The protocol function $d\colon \mathbb{A}\mathrm{gt} \times St \to 2^{Act}$ defines nonempty sets of actions available to agents at each state; we write $d_a(q)$ instead of $d(a, q)$, and define $d_A(q) = \prod_{a\in A} d_a(q)$ for each $A \subseteq \mathbb{A}\mathrm{gt}, q \in St$. Furthermore, o is a (deterministic) transition function that assigns the outcome state $q' = o(q, \alpha_1, \ldots, \alpha_k)$ to each state q and tuple of actions $\langle \alpha_1, \ldots, \alpha_k\rangle$ such that $\alpha_i \in d(i, q)$ for $i = 1, \ldots, k$.

Every $\sim_a \subseteq St \times St$ is an epistemic equivalence relation with the intended meaning that, whenever $q \sim_a q'$, the states q and q' are indistinguishable to agent a. The iCGS is assumed to be *uniform*, in the sense that $q \sim_a q'$ implies $d_a(q) = d_a(q')$. We also define $\sim_A = \bigcup_{a\in A} \sim_a$. The transitive closure of $\sim_A$ is denoted by $\sim_A^C$, and corresponds to *common knowledge* [25] .

Strategies. A strategy of agent $a \in \mathbb{A}\mathrm{gt}$ is a conditional plan that specifies what a is going to do in every possible situation. In this paper, we consider *imperfect information memoryless strategies* [39]. Formally, a strategy for a is a function $\widehat{\sigma}_a\colon St \to Act$ with $\widehat{\sigma}_a(q) \in d_a(q)$ for each $q \in St$ and $\widehat{\sigma}_a(q) = \widehat{\sigma}_a(q')$ for each $q, q' \in St$ such that $q \sim_a q'$. A *collective strategy* $\widehat{\sigma}_A$ for coalition $A \subseteq \mathbb{A}\mathrm{gt}$ is a tuple of individual strategies, one per agent in A.

Partial strategies. A *partial strategy* for a is a partial function $\sigma_a\colon St \rightharpoonup Act$ that can be extended to a strategy. The domain of a partial strategy is denoted by $dom(\sigma_a)$. The set of all partial strategies for $A \subseteq \mathbb{A}\mathrm{gt}$ is denoted by Σ_A.

Let $q \in dom(\sigma_A)$ for some $\sigma_A \in \Sigma_A$. The *outcome* of σ_A from q consists of all the maximal paths $\lambda \in dom(\sigma_A)^* \cup dom(\sigma_A)^\omega$ that follow the partial strategy. Formally we have:

$$\lambda \in out(q, \sigma_A) \text{ iff } \lambda_1 = q \wedge \forall_{i\le|\lambda|}\lambda_i \in dom(\sigma_A)$$
$$\wedge\ \forall_{i<|\lambda|}\exists_{\beta\in d_{\mathbb{A}\mathrm{gt}\setminus A}(\lambda_i)} o(\lambda_i, (\sigma_A(\lambda_i), \beta)) = \lambda_{i+1}$$

where $|\lambda|$ denotes the length (i.e., the number of states) of λ and λ is either infinite (i.e., $|\lambda| = \infty$) or cannot be extended in $out(q, \sigma_A)$. For each $i \in \mathbb{N}$ λ_i is the i–th element in λ and $supp(\lambda)$ is the set of all elements of λ. Moreover, $supp(out(q, \sigma_A)) = \bigcup_{\lambda\in out(q,\sigma_A)} supp(\lambda)$.

Let $Q \subseteq dom(\sigma_A)$. A partial strategy σ_A is *Q-loopless*, if the set $\bigcup_{q\in Q} out(q, \sigma_A)$ contains only finite paths. For each $p \in \mathcal{P}$ we say that σ_A is *p-free* if $V(p) \cap dom(\sigma_A) = \emptyset$.

Finally, we define the *border* of σ_A as the set of states where the partial strategy induces some choices due to uniformity:

$$bord(\sigma_A) = \{q \in St \setminus dom(\sigma_A) \mid \exists_{a\in A}\exists_{q'\in dom(\sigma_A)}\ q' \sim_a q\}.$$

In what follows we often refer to partial strategies simply as strategies and assume a fixed iCGS and $A \subseteq \mathbb{A}\mathrm{gt}$.

3 COMPARING PARTIAL STRATEGIES

In game theory, iterative removal of dominated strategies is a basic method for reducing the space of strategy profiles relevant to solution concepts such as Nash equilibrium. Briefly, if one strategy leads to smaller payoffs than another one, then the former can be omitted in the analysis.

In our formal setting we focus on the property of *enforceability*. We say that $p \in \mathcal{P}$ is enforceable from the state $q \in St$ by a coalition $A \subseteq \mathbb{A}\mathrm{gt}$ using strategy $\sigma_A \in \Sigma_A$ if for each $\lambda \in out(q, \sigma_A)$ there exists $i \in \mathbb{N}$ satisfying $\lambda_i \in V(p)$. This is denoted by $q \models \langle\sigma_A\rangle Fp$. We write $q \models \langle\langle A\rangle\rangle Fp$ if such a strategy exists. Instead of comparing and removing full strategies, we compose a possibly winning strategy from partial strategies. Moreover, we propose a procedure that simplifies winning strategies by substituting their parts.

We say that two partial strategies $\sigma_A, \sigma'_A \in \Sigma_A$ are *conflictless* iff $dom(\sigma_A) \cap dom(\sigma'_A) = \emptyset$ and $\sigma_A \cup \sigma'_A$ is a partial strategy. In order to analyse the interplay between two partial strategies we introduce the following notions.

Definition 3.1. Let $\sigma_A, \sigma'_A \in \Sigma_A$ be conflictless partial strategies. We will use the following notation:

- $O(\sigma_A) = \{q' \in St \setminus dom(\sigma_A) \mid \exists_{q \in dom(\sigma_A)} \exists_{\beta \in d_{\mathbb{A}gt \setminus A}(q)} o(q, (\sigma_A(q), \beta)) = q'\}$, called the *set of exits* of σ_A, consisting of all the states outside of the domain that can be reached by executing the strategy.
- $\mathcal{SC}(\sigma_A, \sigma'_A) = \{q \in bord(\sigma_A) \cup bord(\sigma'_A) \mid \forall_{a \in A} \exists_{q' \in dom(\sigma_A) \cup dom(\sigma'_A)} q \sim_a q'\}$, called the *set of shared control* of both the strategies.
- $\Delta(\sigma_A, \sigma'_A) = \{(q, \alpha) \mid q \in \mathcal{SC}(\sigma_A, \sigma'_A)$ and $\alpha \in d_A(q)$ is induced by σ_A and $\sigma'_A\}$ called the *induced shared-control strategy*.
- $\sigma_A \cup \sigma'_A = \sigma_A \cup \sigma'_A \cup \Delta(\sigma_A, \sigma'_A)$ and $\Lambda(\sigma_A, \sigma'_A) = \sigma_A \cup \Delta(\sigma_A, \sigma'_A)$ that complement $\sigma_A \cup \sigma'_A$ (σ_A, resp.) with shared-control strategy.
- $\mathcal{I}(\sigma_A, \sigma'_A) = (O(\sigma'_A) \cap dom(\Lambda(\sigma_A, \sigma'_A))) \cup q^e_{init}$, called the *set of inputs of* σ'_A into σ_A, where $q^e_{init} = \{q_{init}\}$ if $q_{init} \in dom(\sigma_A)$ and $q^e_{init} = \emptyset$ otherwise. Intuitively, $\mathcal{I}(\sigma_A, \sigma'_A)$ contains all the states that are (1) in the domain of σ_A (2) or jointly controlled by σ_A and σ'_A but not in the domain of σ'_A (3) and s.t. they can be reached by executing σ'_A. For technical convenience, the initial state is added to the inputs into σ_A, if it is present in its domain.

Observe that a coalition $A \subseteq \mathbb{A}gt$ can enforce p from a state q iff it has a strategy that covers q and allows for avoiding loops along each of its executions, at least until p is reached. We thus have the following result.

Proposition 3.2. *We have $q \models \langle\langle A \rangle\rangle Fp$ iff there exists a p-free partial strategy $\sigma_A \in \Sigma_A$ that is q-loopless and such that $q \in dom(\sigma_A)$ and $O(\sigma_A) \subseteq V(p)$.*

3.1 Substituting Strategies

In the rest of this section we aim to formalise the following intuition. Let us assume that two conflictless partial strategies σ^C_A and σ_A control the executions of the system. The joint domain of the strategies is known and the goals that we want to enforce lie outside. For simplicity, let the borders of both strategies be empty. We can list the *inputs* of σ^C_A into σ_A defined as the states reachable by following σ^C_A and controlled by σ_A. To each of these inputs we can assign the set of *outputs*, i.e., the states encountered immediately when leaving the domain of σ_A, after starting from a given input. This assignment is called *input/output characteristic of σ_A w.r.t. σ^C_A*. Recall that the goal is jointly enforced by σ^C_A and σ_A if it is reached by every path that leaves the sum of domains of these strategies. Under this assumption (and other conditions introduced in what follows) we can observe that a partial strategy σ'_A that shares the same set of inputs with σ_A and offers a finer control over the outputs can be substituted for σ_A, when considered w.r.t. σ^C_A.

Definition 3.3 (Input/Output Characteristic). Let $\sigma_A, \sigma^C_A \in \Sigma_A$ be conflictless. The *input/output characteristic of σ_A w.r.t. σ^C_A* is defined as a function $\mathcal{IO}(\sigma_A, \sigma^C_A)\colon \mathcal{I}(\sigma_A, \sigma^C_A) \to 2^{O(\Lambda(\sigma_A, \sigma^C_A))}$ such that for each $q \in \mathcal{I}(\sigma_A, \sigma^C_A)$ we have $\mathcal{IO}(\sigma_A, \sigma^C_A)(q) = O(\Lambda(\sigma_A, \sigma^C_A)) \cap supp(out(q, \Lambda(\sigma_A, \sigma^C_A)))$.

The idea is as follows. For simplicity, let us assume that $\Lambda(\sigma_A, \sigma^C_A) = \sigma_A$, i.e., σ_A, σ^C_A do not share control over any state. Then, $O(\Lambda(\sigma_A, \sigma^C_A))$ contains the states that can be reached by executing σ_A and are outside of the domain of the strategy. Moreover, $supp(out(q, \Lambda(\sigma_A, \sigma^C_A)))$ contains the states of the paths resulting from executing σ_A from q. Thus, $\mathcal{IO}(\sigma_A, \sigma^C_A)(q)$ contains the states that can be reached by executing σ_A from q right after leaving the domain of σ_A. The following example illustrates the definition.

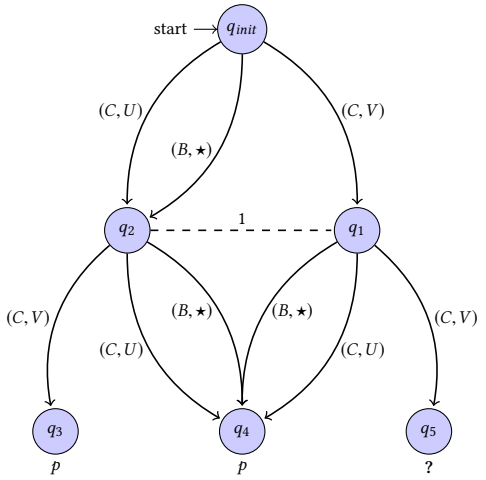

Figure 1: Example iCGS

Example 3.4. Fig. 1 presents an iCGS with six states and two agents. The solid edges denote temporal transitions, and are labeled with pairs of actions selected by the respective agents. The $\star$ symbol indicates that the choice is irrelevant. The dotted line connects states indistinguishable to agent 1.

In state q_{init}, agent 1 can either choose to move to q_2 by playing B or let the other agent make the choice between q_1 and q_2. Note that the states q_1 and q_2 are indistinguishable to agent 1. In $\{q_1, q_2\}$ agent 1 can either fix C or B for both the states in his strategies. In the first case, the second agent can respond by playing U or V, which takes the system to either q_3 or q_4 (when moving from q_2) or to q_4 or q_5 (when moving from q_1). In the second case, irrespectively of the choice of the second player the system moves to q_4. Proposition p holds in states q_3 and q_4. We will fix the labeling of q_5 later.

Let us consider the singleton coalition $A = \{1\}$ and three partial strategies σ^C_A, σ_A, and σ'_A, satisfying $dom(\sigma^C_A) = \{q_{init}\}$ and $dom(\sigma_A) = dom(\sigma'_A) = \{q_2\}$. Let $\sigma^C_A(q_{init}) = C$, $\sigma_A(q_2) = B$, and $\sigma'_A(q_2) = C$. Observe that $O(\sigma^C_A) = \{q_1, q_2\}$ and we have $\Lambda(\sigma_A, \sigma^C_A)(q_i) = B$, for $i \in \{1, 2\}$ and $\Lambda(\sigma'_A, \sigma^C_A)(q_i) = C$. We thus have $\mathcal{I}(\sigma'_A, \sigma^C_A) = \mathcal{I}(\sigma_A, \sigma^C_A) = \{q_1, q_2\}$. Moreover, we also have $\mathcal{IO}(\sigma_A, \sigma^C_A)(q_1) = \mathcal{IO}(\sigma_A, \sigma^C_A)(q_2) = \{q_4\}$, and $\mathcal{IO}(\sigma'_A, \sigma^C_A)(q_1) = \{q_4, q_5\}$ and $\mathcal{IO}(\sigma'_A, \sigma^C_A)(q_2) = \{q_3, q_4\}$.

We now provide a method for comparing partial strategies in the presence of a context, based on their input/output characteristics.

Definition 3.5 (Comparing Partial Strategies). Let $\sigma_A, \sigma'_A, \sigma^C_A \in \Sigma_A$ be s.t. the pairs σ_A, σ^C_A and σ'_A, σ^C_A are conflictless. We call σ^C_A the *context strategy*. We write $\sigma'_A \preceq_{\sigma^C_A} \sigma_A$ iff the following conditions are satisfied:

(1) $dom(\mathcal{IO}(\sigma_A, \sigma_A^C)) = dom(\mathcal{IO}(\sigma'_A, \sigma_A^C))$, i.e., both the partial strategies σ_A and σ'_A control the same set of inputs from the context; and
(2) for each $q \in dom(\mathcal{IO}(\sigma_A, \sigma_A^C))$ we have $\mathcal{IO}(\sigma_A, \sigma_A^C)(q) \subseteq \mathcal{IO}(\sigma'_A, \sigma_A^C)(q)$, i.e., σ_A leads the inputs to exits of σ'_A.

When the context σ_A^C is fixed, relation $\preceq_{\sigma_A^C}$ is a preorder on partial strategies conflictless with σ_A^C. While the strict antisymmetry does not hold in general, observe that $\sigma'_A \preceq_{\sigma_A^C} \sigma_A$ and $\sigma_A \preceq_{\sigma_A^C} \sigma'_A$ imply together that $\mathcal{IO}(\sigma_A, \sigma_A^C) = \mathcal{IO}(\sigma'_A, \sigma_A^C)$.

We now remove the need to specify the context.

Definition 3.6 (Comparison Relation $\preceq$). Define

$$\Gamma_A = \big\{\{\sigma_A, \sigma_A^C\} \subseteq \Sigma_A \mid \sigma_A, \sigma_A^C \text{ are conflictless, } \Lambda(\sigma_A, \sigma_A^C) \text{ is } p\text{-free and } \mathcal{I}(\sigma_A, \sigma_A^C)\text{-loopless and } q_{init} \in dom(\sigma_A) \cup dom(\sigma_A^C)\big\}.$$

The elements of Γ_A are called *strategic coverings*.

Now, let $\gamma_A, \gamma'_A \in \Gamma_A$ be such that $\gamma_A = \{\sigma_A, \sigma_A^C\}$ and $\gamma'_A = \{\sigma'_A, \sigma_A^C\}$. We write $\gamma_A \preceq_0 \gamma'_A$ iff $\sigma'_A \preceq_{\sigma_A^C} \sigma_A$. The relation $\preceq$ is defined as the reflexive and transitive closure of $\preceq_0$.

Note that this relation is also a preorder. A further extension to arbitrary tuples of strategies, instead of pairs, is straightforward, and we omit it for the sake of simplicity.

So far we have introduced notions that allow for comparing strategic coverings. We can now show that the relation of comparison indeed preserves enforceability, which is one of the main contributions in this paper. Intuitively, a stronger strategy (w.r.t. $\preceq$) can achieve at least what the weaker one can. For brevity, we write $\bigcup \gamma_A = \sigma_A^C \cup \sigma_A$ for each $\gamma_A = \{\sigma_A, \sigma_A^C\} \in \Gamma_A$.

Theorem 3.7 (On Substituting Strategies). *Let $\gamma_A, \gamma'_A \in \Gamma_A$ be such that $\gamma'_A \preceq \gamma_A$. If $q_{init} \models \langle\langle \bigcup \gamma'_A \rangle\rangle Fp$, then also $q_{init} \models \langle\langle \bigcup \gamma_A \rangle\rangle Fp$.*

Proof sketch. Let $\gamma'_A = \{\sigma'_A, \sigma_A^C\}$, $\gamma_A = \{\sigma_A, \sigma_A^C\}$, and $\sigma'_A \preceq_{\sigma_A^C} \sigma_A$. Assume that $q_{init} \models \langle\langle \sigma_A^C \cup \sigma'_A \rangle\rangle Fp$ and $\sigma'_A \preceq_{\sigma_A^C} \sigma_A$. If $q_{init} \in dom(\sigma_A^C)$, then a path following $\bigcup \gamma'_A$ from q_{init} cannot loop before reaching p or leaving $dom(\sigma_A^C)$ without visiting a state labeled with p. In the latter case $\Lambda(\sigma'_A, \sigma_A^C)$ takes charge and leads the path from one of its inputs to its exits (there is no other possibility due to its looplessness and p-freeness). Now, it suffices to observe that $\Lambda(\sigma_A, \sigma_A^C)$ performs exactly the same type of transfer between its inputs and exits. The detailed proof and case of $q_{init} \in dom(\sigma'_A)$ is omitted due to lack of space. □

Let us illustrate the theorem with an example.

Example 3.8. Take the scenario in Example 3.4 and Fig. 1. Additionally, let $\sigma'^C_A \in \Sigma_A$ be such that $dom(\sigma'^C_A) = \{q_{init}\}$ and $\sigma'^C(q_{init}) = B$. Observe that $\{\sigma_A^C, \sigma'_A\} \preceq \{\sigma_A^C, \sigma_A\}$, as from Example 3.4 we have $\sigma'_A \preceq_{\sigma_A^C} \sigma_A$. Moreover, it can be calculated that $\sigma_A^C \preceq_{\sigma_A} \sigma'^C_A$, hence $\{\sigma_A^C, \sigma_A\} \preceq \{\sigma'^C_A, \sigma_A\}$. Finally, we have $\{\sigma_A^C, \sigma'_A\} \preceq \{\sigma'^C_A, \sigma_A\}$. Note that the leftmost pair of strategies for agent 1 corresponds to always executing action C, while the rightmost one corresponds to always executing B. The latter is naturally stronger, as it is able to enforce p from q_{init}, even if q_5 is not labeled with p.

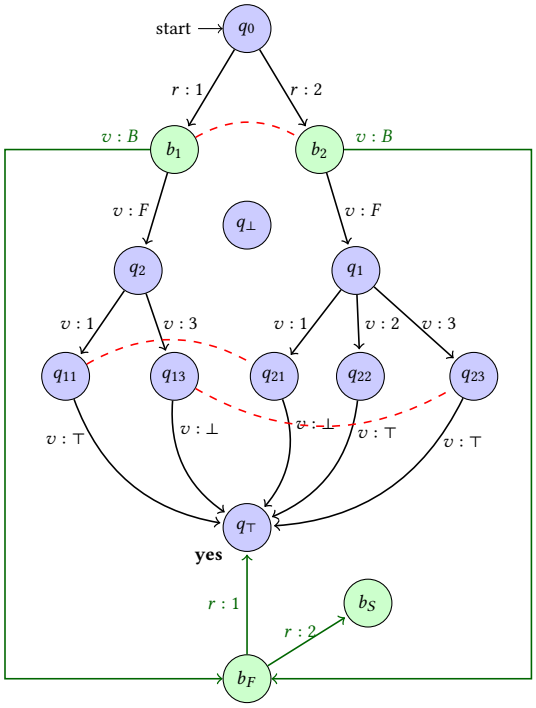

Figure 2: M'_ϕ for $\phi \equiv (x_1 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$. The transitions $(q_{11}, v : \bot, q_\bot)$, $(q_{13}, v : \top, q_\bot)$, $(q_{21}, v : \top, q_\bot)$, $(q_{22}, v : \bot, q_\bot)$, and $(q_{23}, v : \bot, q_\bot)$ are omitted for clarity.

In Section 4, we will provide a practical justification for Theorem 3.7 by proposing an algorithm for on-the-fly synthesis of strategies.

3.2 Complexity

We now tackle the complexity of deciding whether a given strategy is optimal. We write $\sigma_1 \prec_{\sigma_A^C} \sigma_2$ if $\sigma_1 \preceq_{\sigma_A^C} \sigma_2$ but not $\sigma_2 \preceq_{\sigma_A^C} \sigma_1$. A strategy σ_1 is *optimal* w.r.t. a context σ^C if there is no strategy σ_2 s.t. $\sigma_1 \prec_{\sigma_A^C} \sigma_2$.

Theorem 3.9. *Deciding whether a partial strategy is optimal w.r.t. a given context and iCGS is* **co-NP***-complete.*

Proof sketch. We focus on the complement problem, i.e., the question if a strategy σ_1 can be improved w.r.t. a context σ^C over iCGS M. It is not difficult to see that this problem is in **NP**, as we can guess a strategy σ_2, and test in polynomial time if it is uniform, conflictless with σ^C, and whether $\sigma_1 \prec_{\sigma_A^C} \sigma_2$.

To show the **NP**-hardness, we modify the reduction from SAT to enforceability in [27]. There, it is shown how to construct, for any boolean formula ϕ, an iCGS M_ϕ modeling a turn-based game between *the refuter (r)* and *the verifier (v)*. The verifier proposes a strategy that enforces *yes* in M_ϕ iff ϕ is satisfiable. In the first phase of the game, the refuter points to a clause of ϕ. Then, the verifier's strategy is executed by pointing to a literal in the selected

clause and choosing a valuation of the corresponding variable. The indistinguishability relation for v is used to ensure that the choice of valuations is made consistently.

We extend M_ϕ with a gadget to obtain M'_ϕ. Let S denote the set of all states of M'_ϕ. We present the construction in Fig.2 on the case of $\phi \equiv (x_1 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$; the blue states originate from M_ϕ while the green ones belong to the gadget. Firstly, we add a set of states $G = \{b_i\}_{i \in clauses(\phi)}$, one per each clause of ϕ. These states are indistinguishable for the verifier and visited just after the refuter's choice of a clause. Then, the verifier can make a decision of whether to continue (F) following the refuter's choice, or break (B), moving to the new state b_F. The latter is controlled by the refuter who can select to move to the final state $q_\top$ labeled with *yes* or the sink state b_S. Now, let us fix for the verifier the empty context strategy $\sigma^C = \emptyset$ and consider a strategy σ_1 with $dom(\sigma_1) = S \setminus \{q_\top, q_\bot, b_S\}$ and such that $\sigma_1(b) = B$ for all $b \in G$. Note that $\mathcal{I}(\sigma_A, \sigma'_A) = \{q_0\}$, as q_0 is the initial state of M'_ϕ. Intuitively, in σ_1 the verifier decides to break, after which the refuter can choose to move to either $q_\top$ or b_S; the verifier's choices in the remaining states do not influence the outcomes. We thus have $\mathcal{IO}(\sigma_1, \sigma^C)(b) = \{q_\top, b_S\}$ for all $b \in G = \mathcal{I}(\sigma_1, \sigma^C)$.

It now suffices to observe that any strategy σ_2 that improves σ_1 needs to make a choice of action F in the states of G and then enforce $q_\top$ along each path, i.e., enforce *yes* from the initial state. Therefore, σ_1 is improvable w.r.t. σ^C iff *yes* is enforceable. □

3.3 On Independence of Components

In this subsection we lay ground for parallelization of strategy synthesis, that can be explored in the future. Namely, we identify certain conditions under which a strategic covering $\gamma_A = \{\sigma^0_A, \sigma^1_A\}$ can be sent to two independent processes such that: (1) each Process i, where $i \in \{0, 1\}$, computes a maximal strategy $\sigma^{m,i}_A$ dominating σ^i_A w.r.t. the context $\sigma^{i+1(mod2)}_A$; (2) the covering $\gamma^m_A = \{\sigma^{m,1}_A, \sigma^{m,2}_A\}$ dominates γ_A.

Recall that in a preorder $\sqsubset$ an element e is maximal iff for all e' s.t. $e \sqsubset e'$ we also have $e' \sqsubset e$. In Example 3.8 we have identified a pair of strategies maximal w.r.t. relation $\preceq$ in iCGS of Fig. 1, namely $\{\sigma_A, \sigma'^C_A\}$. We also say that a strategy $\sigma_A \in \Sigma_A$ is σ^C_A-*independent* if there are no states $q \in dom(\Lambda(\sigma_A, \sigma^C_A))$, $q' \in dom(\sigma^C_A)$, and agent $a \in A$ such that $q \sim_a q'$.

The meaning of the following lemma is that a change of the context strategy does not affect the relation between two strategies if the context stays epistemically independent from them and does not produce new entry points.

Lemma 3.10 (On Shrinking Contexts). *Let $\sigma_A, \sigma'_A, \sigma^C_A, \sigma^{Cm}_A \in \Sigma_A$ be such strategies that σ_A and σ'_A are σ^C_A- and σ^{Cm}_A-independent. If $O(\sigma^{Cm}_A) \subseteq O(\sigma^C_A)$ and $\sigma'_A \preceq_{\sigma^C_A} \sigma_A$, then also $\sigma'_A \preceq_{\sigma^{Cm}_A} \sigma_A$.*

Proof sketch. Observe that $\sigma'_A \preceq_{\sigma^C_A} \sigma_A$ requires $\mathcal{I}(\sigma'_A, \sigma^C_A) = \mathcal{I}(\sigma_A, \sigma^C_A)$. This, together with σ^C_A- and σ^{Cm}_A-independence of σ_A and σ'_A yields $\mathcal{I}(\sigma'_A, \sigma^{Cm}_A) = \mathcal{I}(\sigma'_A, \sigma^C_A) \cap O(\sigma^{Cm}_A) = \mathcal{I}(\sigma_A, \sigma^C_A) \cap O(\sigma^{Cm}_A) = \mathcal{I}(\sigma_A, \sigma^{Cm}_A)$. As $\mathcal{IO}(\sigma_A, \sigma^{Cm}_A)$ and $\mathcal{IO}(\sigma'_A, \sigma^{Cm}_A)$ are restrictions of $\mathcal{IO}(\sigma_A, \sigma^C_A)$ and $\mathcal{IO}(\sigma'_A, \sigma^C_A)$, resp., to a common domain, and $\sigma'_A \preceq_{\sigma^C_A} \sigma_A$, we obtain $\sigma'_A \preceq_{\sigma^{Cm}_A} \sigma_A$. □

As a particular case we can apply Lemma 3.10 to strategies defined on classes of the relation of common knowledge of A. Namely, it suffices to observe that for each $\sigma_A, \sigma^C_A \in \Sigma_A$ if $[dom(\sigma_A)]_{\sim^C_A} \cap [dom(\sigma^C_A)]_{\sim^C_A} = \emptyset$, then σ_A is σ^C_A-*independent* and vice versa. This suggests a natural initial partitioning scheme, where partial strategies are defined on the abstraction classes of $\sim^C_A$. We note in passing the analogy to some existing model equivalences and approximation schemes for $\mathbf{ATL}_{\mathrm{ir}}$ [4, 28].

The next theorem points to some sources of conflict between two strategies, leading to a natural observation that a strategy can be safely replaced with a stronger one, if the observed change is not visible outside of its domain.

Theorem 3.11 (Component-wise Comparison). *Let $\gamma_A, \gamma^m_A \in \Gamma_A$ be s.t. $\gamma_A = \{\sigma_A, \sigma^C_A\}$ and $\gamma^m_A = \{\sigma^m_A, \sigma^{Cm}_A\}$. If both the conditions hold:*

C1 *$\sigma_A \preceq_{\sigma^C_A} \sigma^m_A$ and $\{\sigma^m_A, \sigma^C_A\} \in \Gamma_A$ and $O(\sigma^m_A) \subseteq O(\sigma_A)$,*

C2 *$\sigma^C_A \preceq_{\sigma_A} \sigma^{Cm}_A$ and $\{\sigma_A, \sigma^{Cm}_A\} \in \Gamma_A$ and $\sigma^C_A, \sigma^{Cm}_A$ are σ_A, σ^m_A-independent,*

then $\gamma_A \preceq \gamma^m_A$.

Proof. As $\sigma_A \preceq_{\sigma^C_A} \sigma^m_A$, it follows from the definition of $\preceq$ that $\{\sigma_A, \sigma^C_A\} \preceq \{\sigma^m_A, \sigma^C_A\}$. Now, from Lemma 3.10 applied to $\sigma^C_A \preceq_{\sigma_A} \sigma^{Cm}_A$ under combined conditions **C1** and **C2** we have $\sigma^C_A \preceq_{\sigma^m_A} \sigma^{Cm}_A$, hence $\{\sigma^m_A, \sigma^C_A\} \preceq \{\sigma^m_A, \sigma^{Cm}_A\}$. We obtain $\gamma_A \preceq \gamma^m_A$ from the transitivity of $\preceq$. □

This result suggests a possible optimization of our method, that we plan to explore in future work.

4 EVALUATION

In this section we show how to apply the concept of strategic dominance to two tasks: model checking of enforceability and optimization of an existing strategy. We evaluate our framework on several scalable benchmarks.

4.1 Setup of the Experiments

The dominance-based algorithms have been implemented in Python 3. We use non-symbolic representations, i.e., the models are stored in memory explicitly, as transition graphs. We compare our new algorithms to the output of three existing tools: the state of the art tool MCMAS [31], an experimental model checker SMC [37], and a prototype implementation (in C++) of the fixpoint approximation algorithms of [29]. All the tests have been conducted on a laptop with an Intel Core i7-6700HQ CPU with dynamic clock speed of 2.60 GHz up to 3.50 GHz, 32 GB RAM, and 64bit Linux. The running times are given in seconds; the timeout was 4h.

Heuristics. While comparing the input/output characteristics provides a sufficient condition for preserving goal-enforceability, in practice simple plans are usually preferable. Informally, a plan is simpler than another if it is easier to remember and execute. To reflect this, we define heuristic preorders on strategies that express different kinds of domain-specific relationships. E.g., a strategy σ_A can be seen as simpler than σ'_A if it utilises less actions, its branching factor is smaller, more nodes are fully controlled by A,

Algorithm 1 DominoDFS($\sigma_A, p, \preceq_{\mathcal{H}}$)

Input: σ_A - a p-free and initial state-loopless partial strategy for A, proposition p, and heuristic preorder $\preceq_{\mathcal{H}}$
Output: an extension of σ_A that enforces p from the initial state q_{init} or $\emptyset$ is there is none

1: **if** $\sigma_A = \emptyset$ **then**
2: $nxtstate = q_{init}$ {Initialisation}
3: **else if** exists $x \in O(\sigma_A)$ s.t. $x \notin V(p)$ **then**
4: $nxtstate = x$ {σ_A is not winning and extension is possible}
5: **else**
6: **return** σ_A {σ_A is a winning strategy}
7: **end if**
8:
9: $candactions = d_A(nxtstate)$
10: $candactions.removeDominatedStrategies(context = \sigma_A)$
11: $candactions.removeLoops(baseStrategy = \sigma_A)$
12: **if** $candactions = \emptyset$ **then**
13: **return** $\emptyset$ {No winning extension of σ_A possible}
14: **end if**
15: $candactions.heuristicOrdering(\preceq_{\mathcal{H}})$
16:
17: **for each** $actn \in candactions$ **do**
18: $solution =$ **DominoDFS**$(\sigma_A \cup \{(nxtstate, actn)\}, p, \preceq_{\mathcal{H}})$
19: **if** $solution \neq \emptyset$ **then**
20: **return** $solution$
21: **end if**
22: **end for**
23:
24: **return** $\emptyset$ {No winning extension of σ_A possible}

etc. We denote such preorders on Σ_A by $\preceq_{\mathcal{H}}$ and assume that they are extended to Γ_A in a natural way.

In what follows, we use three different approaches. The first one, called the *Simple Reduction*, is based only on the preorder $\preceq$. The second one, called the *Epistemic Heuristic*, employs a heuristic preorder that compares the size of the sets of exits of two partial strategies (including indistinguishable states); the better one contains less states. The third one, the *Control Heuristic*, deems one partial strategy better than another if its exits contain less non-controlled states. Intuitively, a state q is controlled by coalition A if A fully controls the next transition. Formally, q *is controlled by* A iff $d_{\mathbb{A}gt\setminus A}(q)$ is a singleton.

4.2 Depth-First Strategy Search: DominoDFS

Our main procedure for dominance-based strategy synthesis is presented in Algorithm 1. The goal of the algorithm is to synthesise a strategy that enforces p and extends σ_A from the initial state of the model. Additionally, the algorithm uses a heuristic preorder $\preceq_{\mathcal{H}}$. As previously, the model is implicit and omitted for clarity.

The requirements on the input strategy σ_A given in the algorithm are also its invariant. Lines 1–7 establish whether it is possible to find a state *nxtstate* that is present in any extension of σ_A. If not, then by the assumption on σ_A and Proposition 3.2 the strategy is winning and is returned. Otherwise, the allowed single-step actions for A in the state *nxtstate* are collected in the list *candactions* (Line 9).

Conf.	DominoDFS	MCMAS	Approx.	Approx. opt.
(1, 1)	0.0006	0.12	0.0008	< 0.0001
(2, 2)	0.01	8712*	0.01	< 0.0001
(3, 3)	0.8	timeout	0.8	0.06
(4, 4)	160	timeout	384	5.5
(5, 5)*	1373	timeout	8951	39
(5, 5)	memout	timeout	memout	138
(6, 6)*	memout	timeout	memout	4524

Table 1: Results for Bridge

Conf.	DominoDFS	MCMAS	SMC
(1, 1, 1)	0.3	65	63
(2, 1, 1)	1.5	12898	184
(3, 1, 1)	25	timeout	6731
(2, 2, 1)	25	timeout	4923
(2, 2, 2)	160	timeout	timeout
(3, 2, 2)	2688	timeout	timeout
(3, 3, 2)	timeout	timeout	timeout

Table 2: Results for Castles

Then, the list is pruned by removing all the single-step actions dominated w.r.t. the context σ_A (Line 10) and those that induce loops when added to σ_A (Line 11). If no action is left in *candactions*, then it is not possible to build a winning strategy based on σ_A. Otherwise the list *candactions* is ordered to respect $\preceq_{\mathcal{H}}$ (Line 15); note that if the heuristic preorder is a total order then this step boils down to simply sorting w.r.t. $\preceq_{\mathcal{H}}$. The algorithm is then recursively called on all the candidate extensions of σ_A (Line 20) until a solution is found or no candidates are left (Line 24).

4.2.1 First Benchmark: Bridge Endplay [29]. We use a Bridge play scenario of a type often considered in bridge handbooks and magazines. There are four players, depicted by the four sides of the world. The task is to find a winning strategy for the declarer located at the South position, in the k-endplay of the game. The deck consists of $4n$ cards in total (n in each suit), and the initial state captures each player holding k cards in their hand, after having played $n-k$ cards. This way we obtain a family of models, parameterized by the possible values of (n, k). A NoTrump contract is being played; the declarer wins if she takes more than $k/2$ tricks in the endplay. The verified property $\phi_{\text{Bridge}} \equiv \langle\langle S \rangle\rangle F\, win$ asks if the declarer has a strategy to collecting more tricks than the enemy team.

4.2.2 Second Benchmark: Castles [37]. The model describes a scenario of war between three castles. Each castle has a certain amount of health points and a number of worker agents. Each worker is employed by a single castle and performs one of the four actions: defend the castle, attack one of the other castles or wait. The agents must rest after defending a castle, so they cannot perform the defend action twice in a row. The goal of the game is to defeat the other castles. A castle is defeated when its health points drop to or below zero. The health decreases during a battle if the number of attackers is greater than the number of defenders, and the damage is equal to the difference. We verify the formula

Algorithm 2 SimplifyStrat$(\gamma_A, \preceq_{\mathcal{H}}, \mu)$

Input: $\gamma_A \in \Gamma_A$, heuristic preorder $\preceq_{\mathcal{H}}$, and quality condition μ
Output: $\gamma_A^m \in \Gamma_A$ satisfying $\gamma_A(\preceq \cap \preceq_{\mathcal{H}})\gamma_A^m$ and $\mu(\gamma_A^m)$

1: **for each** $\pi \in Sym(\gamma_A)$ **do**
2: **for each** $1 \leq i \leq |\pi|$ **do**
3: $\sigma^m_{A,i} :=$ **SingleMax**$(\pi_i, \bigcup_{j \neq i} \pi_j, \preceq_{\mathcal{H}})$
4: **Update** $\pi_i := \sigma^m_{A,i}$
5: **end for**
6: **if** $\mu(\pi)$ **then**
7: **return** π
8: **end if**
9: **end for**

$\phi_{\text{Castles}} \equiv \langle\langle c12 \rangle\rangle F castle3Defeated$, i.e., whether the coalition of the workers from castles 1 and 2 can defeat the third castle.

4.2.3 Experimental Results and Discussion. Table 1 collects the experimental results for models of the Bridge scenario. The first column identifies a subclass of the models. For each such subclass, we have run tests on 50 randomly generated card deals, except for the configurations marked with (*) where we were only able to run tests on a single handcrafted instance of the model due to timeout or memout. The other columns present the average performance of model checking (model generation + verification time): first for our new algorithm (DominoDFS), and then for the reference tools. Due to limitations of SMC, we have not been able to properly encode the benchmarks in that tool, hence we only provide a comparison to MCMAS and the fixed-point approximations. The approximations are used in two variants: the basic one (Approx.) and the optimized one (Approx. opt.), cf. [29] for details.

Table 2 collects the experimental results for Castles. Each triple in the Configuration column refers to the number of workers assigned to the corresponding castles. The initial amount of health points for each castle is 3. All the remaining details are as in Table 1. Since the benchmark does not fulfil the necessary condition for fixpoint approximation [29], we only compare the performance of the dominance-based synthesis to MCMAS and SMC. The times are given in seconds, and the timeout is 4 hours.

The results show that DominoDFS significantly outperforms MCMAS and SMC. It also successfully competes with the basic implementation of fixpoint approximation. We also note that our new approach can handle models that do not submit to the fixpoint approximation scheme. This allows us to suggest the following mete-procedure for verification of enforceability in models with incomplete information: first try optimized fixed-point approximations, if this fails then apply DominoDFS, finally try your luck with remaining tools.

Somewhat surprisingly, none of the heuristics have performed notably better or worse than the simple reduction, hence we omit their performance from the tables.

4.3 Strategy Optimization

In this section we provide an evaluation of our approach applied to optimization of an input strategy. The idea is to start with a winning strategy and consecutively reduce it until it meets additional requirements on simplicity. To this end, we assume the existence of a predicate μ s.t. $\mu(\gamma_A^m)$ holds if the strategic covering γ_A^m is of acceptable quality.

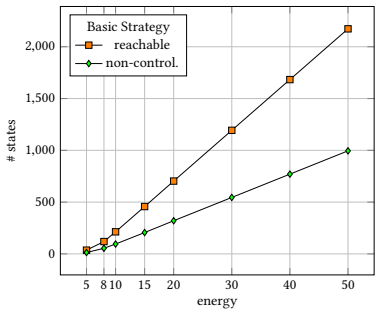

Figure 3: Basic Strategy (Single Drone)

Moreover, for each $\gamma_A \in \Gamma_A$, let $Sym(\gamma_A)$ denote the set of all permutations of components in γ_A. Note that strategic coverings in this section consist of arbitrary amounts of elements. For each $\pi \in Sym(\gamma_A)$ and $1 \leq i \leq |Sym(\gamma_A)|$, let π_i denote its i–th component. With a slight abuse of notation, we sometimes treat a permutation as a writable vector of the components.

The strategy optimization procedure is presented in Algorithm 2. The goal is to simplify the input strategic covering $\gamma_A \in \Gamma_A$ w.r.t. a preorder heuristic $\preceq_{\mathcal{H}}$ and a quality condition μ. The algorithm selects a permutation $\pi \in Sym(\gamma_A)$ and maximises it component-by-component. An auxiliary procedure **SingleMax**$(\sigma'_A, \sigma^C_A, \preceq_{\mathcal{H}})$ returns a maximal strategy σ^m_A satisfying $\{\sigma'_A, \sigma^C_A\} \preceq \cap \preceq_{\mathcal{H}} \{\sigma^m_A, \sigma^C_A\}$, for each $\sigma'_A, \sigma^C_A \in \Sigma_A$. Note that this choice is nondeterministic; a more exhaustive version iterates over all such maximal strategies. Then, a new tuple of strategies ρ is tested and returned if it meets the criteria of μ.

We note in passing that each run of the outer loop 1–9 is independent, hence parallelizable. The runs of the inner loop 2–5 can be executed concurrently if the conditions of Theorem 3.11 hold. This suggests a possible improvement of the implementation by parallel execution, that we plan to explore in the future.

4.3.1 Benchmark: Drone-Control Model. We evaluate the algorithm on models of a team of drones equipped with an initial number of energy points, inspecting a fixed, two-dimensional map. Each action of a drone consumes an energy point. Once the energy level reaches zero, the drone becomes useless. The limited precision of the drone's telemetry is modeled by making selected locations indistinguishable. The drone can attempt to fly in one of the four directions (NESW) unless there is an obstacle. It can also wait, staying in the current place. In some locations, the outcome of its actions can be influenced by the wind that alters the direction of the movement. The states that contain those locations are called *non-controlled*. The action of waiting is thus active: a drone stays in its current location, opposing the wind. It can also decide to drift with the wind, executing the action Fly.

4.3.2 Experimental Results and Discussion. We consider two subclasses of the drone benchmark: one with a single drone, and one with two drones inspecting the same map. The model is scalable

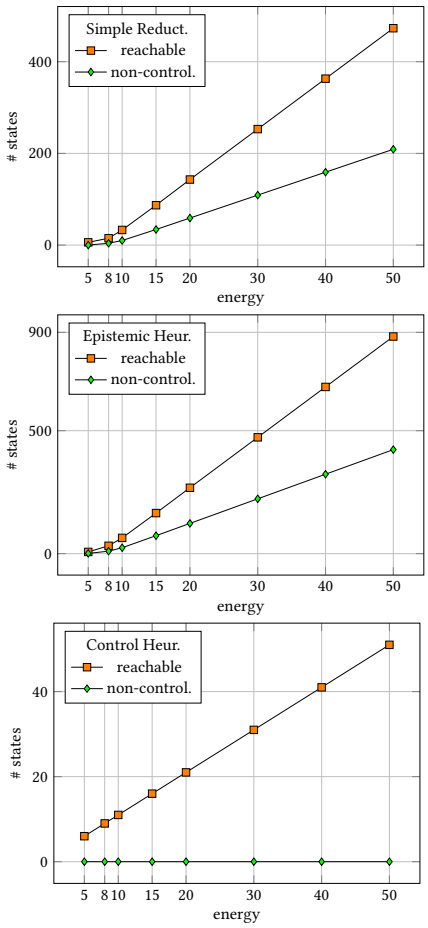

Figure 4: Reduced Strategies (Single Drone)

w.r.t. the number of initial energy points. The basic strategy to be optimized is always the same: drift with the wind (if detected) or wait, being passively moved by the environment until the battery runs out. The characteristics of the basic strategy is presented in Fig. 3. The boxed line shows the number of *reachable states*, i.e., the states that must be taken into account when the strategy is executed (including the states reachable by epistemic indistinguishability links). The diamond line indicates the number of *reachable uncontrollable states*, i.e., reachable states where the wind is present.

The experimental results for the single-drone model scaled w.r.t. the energy points of the drone are presented in Fig. 4. As it can be observed, the simple approach allows for substantial reductions of the basic strategy, leading to 50%–75% smaller sets of outcome states. Rather surprisingly, the epistemic heuristic leads to smaller improvements (only up to 50%). In contrast, the control heuristic provides the best reductions: the space of reachable states becomes up to *50 times smaller*. Moreover, the output strategy traverses only fully controllable states, i.e., the number of non-controllable reachable states *goes down to 0*.

The final batch of experiments was designed to further inspect the efficiency of our approach. To this end, we used the two-drone

Basic strat.		Simple Reduct.		Epistemic Heur.		Control Heur.	
reach	nctr	%reach	%nctr	%reach	%nctr	%reach	%nctr
782	256	1	2	1	2	5	8
1022	350	4	6	1	2	1	2
1147	385	2	4	5	6	3	4
1257	430	10	13	9	13	5	9
1601	512	3	5	3	6	2	4
1853	625	4	5	4	4	4	4
2527	834	1	2	1	2	1	2

Table 3: Randomised Example (2 Drones / 5 Energy Pts)

variant of the Drone benchmark, and randomly transformed approximately 50% of locations by adding non-controllable actions (i.e., the wind). The output of the experiments is presented in Table 3, with the percentages showing the fraction of reachable states that are left after the optimization. The results consistently display a high degree of reduction: the optimized strategies have roughly 100 times smaller sets of reachable states (as well as non-controllable reachable states) regardless of the heuristics being used.

5 CONCLUSIONS

In this paper, we present a framework for strategy synthesis and optimization based on a new notion of strategic dominance w.r.t. a given context. The key idea is to identify the critical parts of execution of the system, as controlled by a given partial strategy. In our case, we have selected, as the primitive building block, the correspondence between each state where the strategy gains the control and the set of states where the strategy returns the control to the environment.

Based on the concept, we prove that strategic dominance preserves enforceability. We identify a related decision problem of optimality for partial strategies, and show that it is **co-NP**-complete. Moreover, we develop a framework for on-the-fly model checking of strategic abilities through strategy synthesis, that uses strategic dominance to reduce the space of candidate strategies. We also present how to apply the dominance to optimize an existing strategy with respect to a given quality condition and a heuristic preorder. Finally, we evaluate the framework experimentally on several scalable benchmarks with very promising results.

A number of interesting research paths is left for future work. Firstly, there is a vast room for improvement for the presented algorithms, e.g., by means of efficient data structures or more successful heuristics. Secondly, we have shown that there are certain conditions under which the synthesis can be performed in a fully parallel way. We plan to refine the conditions, and come up with parallel algorithms for strategy synthesis. Note that, even if fully independent partitioning of state-space is not possible, robust communication and conflict resolution can significantly improve the efficiency. Thirdly, and perhaps most importantly, the presented theory is not suitable for dealing with properties other than enforceability. We thus plan to extend it to handle a wider palette of formulas, ideally targeting the full language of $\mathbf{ATL}_{\mathrm{ir}}$.

Acknowledgements. The authors acknowledge the support of the National Centre for Research and Development (NCBR), Poland, under the PolLux project VoteVerif (POLLUX-IV/1/2016). We also thank the anonymous reviewers for their useful comments.

REFERENCES

[1] N. Alechina, B. Logan, N.H. Nga, and A. Rakib. 2009. A Logic for Coalitions with Bounded Resources. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*. 659–664.

[2] N. Alechina, B. Logan, H.N. Nguyen, and A. Rakib. 2010. Resource-Bounded Alternating-Time Temporal Logic. In *Proceedings of International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 481–488.

[3] R. Alur, T. A. Henzinger, and O. Kupferman. 2002. Alternating-Time Temporal Logic. *J. ACM* 49 (2002), 672–713.

[4] F. Belardinelli, R. Condurache, C. Dima, W. Jamroga, and A.V. Jones. 2017. Bisimulations for Verification of Strategic Abilities with Application to ThreeBallot Voting Protocol. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2017*. IFAAMAS, 1286–1295.

[5] F. Belardinelli and A. Lomuscio. 2017. Agent-based Abstractions for Verifying Alternating-time Temporal Logic with Imperfect Information. In *Proceedings of International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. ACM, 1259–1267.

[6] M. Benedetti, A. Lallouet, and J. Vautard. 2008. Quantified Constraint Optimization. In *Principles and Practice of Constraint Programming, 14th International Conference, CP 2008, Sydney, Australia, September 14-18, 2008. Proceedings (Lecture Notes in Computer Science)*, Vol. 5202. Springer, 463–477.

[7] J. Bernet, D. Janin, and I. Walukiewicz. 2002. Permissive strategies: from parity games to safety games. *ITA* 36, 3 (2002), 261–275.

[8] R. Berthon, B. Maubert, A. Murano, S. Rubin, and M. Y. Vardi. 2017. Strategy Logic with Imperfect Information. In *Proceedings of LICS*. 1–12.

[9] P. Bouyer, N. Markey, J. Olschewski, and M. Ummels. 2011. Measuring Permissiveness in Parity Games: Mean-Payoff Parity Games Revisited. In *Automated Technology for Verification and Analysis, 9th International Symposium, ATVA 2011, Taipei, Taiwan, October 11-14, 2011. Proceedings (Lecture Notes in Computer Science)*, Vol. 6996. Springer, 135–149.

[10] T. Brazdil, K. Chatterjee, M. Chmelik, A. Fellner, and J. Kretinsky. 2015. Counterexample Explanation by Learning Small Strategies in Markov Decision Processes. In *Proceedings of CAV*. 158–177.

[11] T. Brazdil, K. Chatterjee, J. Kretinsky, and V. Toman. 2018. Strategy Representation by Decision Trees in Reactive Synthesis. In *Proceedings of TACAS*. To appear.

[12] R. E. Bryant. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. on Computers* 35, 8 (1986), 677–691.

[13] N. Bulling, J. Dix, and W. Jamroga. 2010. Model Checking Logics of Strategic Ability: Complexity. In *Specification and Verification of Multi-Agent Systems*, M. Dastani, K. Hindriks, and J.-J. Meyer (Eds.). Springer, 125–159.

[14] N. Bulling and B. Farwer. 2010. Expressing Properties of Resource-Bounded Systems: The Logics RTL* and RTL. In *Proceedings of Computational Logic in Multi-Agent Systems (CLIMA) (Lecture Notes in Computer Science)*, Vol. 6214. 22–45.

[15] N. Bulling and B. Farwer. 2010. On the (Un-)Decidability of Model Checking Resource-Bounded Agents. In *Proceedings of ECAI (Frontiers in Artificial Intelligence and Applications)*, Vol. 215. IOS Press, 567–572.

[16] N. Bulling and W. Jamroga. 2011. Alternating Epistemic Mu-Calculus. In *Proceedings of IJCAI-11*. 109–114.

[17] Jerry R. Burch, Edmund M. Clarke, Kenneth L. McMillan, David L. Dill, and L. J. Hwang. 1990. Symbolic Model Checking: 10^20 States and Beyond. In *Proc. of 4th Ann. IEEE Symp. on Logic in Computer Science (LICS)*. IEEE Computer Society, 428–439.

[18] S. Busard. 2017. *Symbolic Model Checking of Multi-Modal Logics: Uniform Strategies and Rich Explanations.* Ph.D. Dissertation. Universite Catholique de Louvain.

[19] S. Busard, C. Pecheur, H. Qu, and F. Raimondi. 2015. Reasoning about memoryless strategies under partial observability and unconditional fairness constraints. *Information and Computation* 242 (2015), 128–156.

[20] J. Calta, D. Shkatov, and B.-H. Schlingloff. 2010. Finding Uniform Strategies for Multi-agent Systems. In *Proceedings of Computational Logic in Multi-Agent Systems (CLIMA) (Lecture Notes in Computer Science)*, Vol. 6245. Springer, 135–152.

[21] P. Cermak, A. Lomuscio, F. Mogavero, and A. Murano. 2014. MCMAS-SLK: A Model Checker for the Verification of Strategy Logic Specifications. In *Proc. of Computer Aided Verification (CAV) (Lecture Notes in Computer Science)*, Vol. 8559. Springer, 525–532.

[22] P. Dembiński, A. Janowska, P. Janowski, W. Penczek, A. Półrola, M. Szreter, B. Woźna, and A. Zbrzezny. 2003. Verics: A Tool for Verifying Timed Automata and Estelle Specifications. In *Proceedings of the of the 9th Int. Conf. on Tools and Algorithms for Construction and Analysis of Systems (TACAS'03)*. Lecture Notes in Computer Science, Vol. 2619. Springer, 278–283.

[23] C. Dima, B. Maubert, and S. Pinchinat. 2015. Relating Paths in Transition Systems: The Fall of the Modal Mu-Calculus. In *Proceedings of Mathematical Foundations of Computer Science (MFCS) (Lecture Notes in Computer Science)*, Vol. 9234. Springer, 179–191.

[24] L. Doyen and J.-F. Raskin. 2011. Games with Imperfect Information: Theory and Algorithms. In *Lecture Notes in Game Theory for Computer Scientists*. Cambridge University Press, 185–212.

[25] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. 1995. *Reasoning about Knowledge*. MIT Press.

[26] P. Gammie and R. van der Meyden. 2004. MCK: Model Checking the Logic of Knowledge. In *Proc. of the 16th Int. Conf. on Computer Aided Verification (CAV'04) (LNCS)*, Vol. 3114. Springer-Verlag, 479–483.

[27] W. Jamroga and J. Dix. 2006. Model Checking ATL$_{ir}$ is Indeed Δ_2^P-complete. In *Proceedings of European Workshop on Multi-Agent Systems (EUMAS) (CEUR Workshop Proceedings)*, Vol. 223. CEUR-WS.org.

[28] W. Jamroga, M. Knapik, and D. Kurpiewski. 2017. Fixpoint Approximation of Strategic Abilities under Imperfect Information. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2017*. IFAAMAS, 1241–1249.

[29] W. Jamroga, M. Knapik, D. Kurpiewski, and Łukasz Mikulski. 2018. Approximate Verification of Strategic Abilities under Imperfect Information. *Artificial Intelligence* (2018). To appear.

[30] W. Jamroga, V. Malvone, and A. Murano. 2017. Reasoning about Natural Strategic Ability. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2017*. IFAAMAS, 714–722.

[31] A. Lomuscio, H. Qu, and F. Raimondi. 2015. MCMAS: An Open-Source Model Checker for the Verification of Multi-Agent Systems. *International Journal on Software Tools for Technology Transfer* (2015).

[32] A. Lomuscio and F. Raimondi. 2006. Model Checking Knowledge, Strategies, and Games in Multi-Agent Systems. In *Proceedings of International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 161–168.

[33] V. Malvone, A. Murano, and L. Sorrentino. 2018. Additional Winning Strategies in Reachability Games. *Fundam. Inform.* 159, 1-2 (2018), 175–195.

[34] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. 2014. Reasoning About Strategies: On the Model-Checking Problem. *ACM Transactions on Computational Logic* 15, 4 (2014), 1–42.

[35] F. Mogavero, A. Murano, and M.Y. Vardi. 2010. Reasoning About Strategies. In *Proceedings of FSTTCS*. 133–144.

[36] D. Neider. 2011. Small Strategies for Safety Games. In *Automated Technology for Verification and Analysis, 9th International Symposium, ATVA 2011, Taipei, Taiwan, October 11-14, 2011. Proceedings (Lecture Notes in Computer Science)*, Vol. 6996. Springer, 306–320.

[37] J. Pilecki, M.A. Bednarczyk, and W. Jamroga. 2017. SMC: Synthesis of Uniform Strategies and Verification of Strategic Ability for Multi-Agent Systems. *Journal of Logic and Computation* 27, 7 (2017), 1871–1895.

[38] F. Raimondi and A. Lomuscio. 2007. Automatic Verification of Multi-agent Systems by Model Checking via Ordered Binary Decision Diagrams. *J. Applied Logic* 5, 2 (2007), 235–251.

[39] P. Y. Schobbens. 2004. Alternating-Time Logic with Imperfect Recall. *Electronic Notes in Theoretical Computer Science* 85, 2 (2004), 82–93.