# Pretty Good Strategies and Where to Find Them

Wojciech Jamroga[1,2] and Damian Kurpiewski[2,3]

[1] Interdisciplinary Centre for Security, Reliability and Trust, SnT, University of Luxembourg, Luxembourg
[2] Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland
[3] Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, Toruń, Poland

**Abstract.** Synthesis of bulletproof strategies in imperfect information scenarios is a notoriously hard problem. In this paper, we suggest that it is sometimes a viable alternative to aim at "reasonably good" strategies instead. This makes sense not only when an ideal strategy cannot be found due to the complexity of the problem, but also when no winning strategy exists at all. We propose an algorithm for synthesis of such "pretty good" strategies. The idea is to first generate a surely winning strategy with *perfect information*, and then iteratively improve it with respect to two criteria of dominance: one based on the amount of conflicting decisions in the strategy, and the other related to the tightness of its outcome set. We focus on reachability goals and evaluate the algorithm experimentally with very promising results.

**Keywords:** Strategy synthesis · imperfect information · alternating-time temporal logic · model checking

## 1 Introduction

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. For example, functionality requirements can be often understood in terms of the user's ability to complete the selected tasks. Similarly, many security properties boil down to inability of the intruder to obtain his goals. *Logics of strategic reasoning* provide powerful tools to reason about such aspects of MAS [3, 45, 40, 7, 8, 29]. A typical property that can be expressed says that *the group of agents $A$ has a collective strategy to enforce temporal property $\varphi$, no matter what the other agents in the system do.* In other words, $A$ have a "winning strategy" that achieves $\varphi$ 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 [1, 22, 27, 18, 19, 37, 34, 36]. Moreover, model checking of

strategic formulas typically relies on synthesis of a suitable strategy to demonstrate that such a strategy exists.

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 [10, 26, 39, 7, 8]. This can be overcome to some degree by using efficient symbolic methods and data structures [9, 13, 27, 44]. However, real-life agents typically have limited capabilities of observation, action, and reasoning. That brings additional challenges. First, the theoretical complexity of model checking for imperfect information strategies (sometimes called *uniform strategies*) ranges from **NP**–complete to undecidable [45, 10, 25], depending on the precise setup of the problem. Secondly, practical attempts at verification suffer from state-space and transition-space explosion. Thirdly, there is no simple fixed-point characterisation of typical properties [24, 12]. As a consequence of the latter, most approaches to synthesis and verification boil down, in the worst case, to checking all the possible strategies [38, 16, 17, 43, 35]. Unfortunately, the strategy space is huge – usually larger than the state space by orders of magnitude, which makes brute-force search hopeless.

An interesting attempt at heuristic search through the strategy space has been proposed in [35]. There, a concept of domination between strategies was introduced, based on the "tightness" of the outcome sets induced by the strategies. Formally, strategy $s$ dominates $s'$ if the set of possible executions of $s$ is a strict subset of the executions of $s'$. The intuition is that those strategies are better which give the agent a better grip on what is going to happen, and better reduce the nondeterminism of the system. Then, the authors of [35] proposed an algorithm for synthesis of uniform strategies, based on depth-first search through the strategy space with simultaneous optimization of dominated partial strategies. The algorithm, dubbed DominoDFS, performed with considerable success on several benchmarks. This might have had two related reasons. First, restricting the set of successor states reduces the possibility of encountering a "bad" successor further on. Perhaps even more importantly, it reduces the space of reachable states, and hence has the potential to considerably speed up the computation.

In this paper, we take the idea of dominance-based optimization, and apply it from a completely different angle. Most importantly, we propose that searching for a "reasonably good" strategy is sometimes a viable alternative to the search for an ideal one (where "ideal" means a surely winning imperfect information strategy). This obviously makes sense when no winning strategy exists, but also when an ideal strategy cannot be found due to the complexity of the problem. Moreover, we propose a procedure for synthesis of such "pretty good" strategies. The algorithm starts with generating a surely winning strategy with *perfect information*. Then, it iteratively improves it with respect to two criteria of dominance: one based on the amount of conflicting decisions in the strategy, and the other related to the tightness of its outcome set. It is worth noting that this is an *anytime* algorithm. Thus, it *always* returns some strategy, provided that a perfect information strategy has been generated in the first phase.

We evaluate the algorithm experimentally on randomly generated concurrent game structures with imperfect information, as well as the scalable Drones benchmark of [35]. The results are compared to the output of DominoDFS and to the fixpoint approximation algorithm of [33], forming a very promising pattern. In particular, for models with relatively small information sets (a.k.a. epistemic indistinguishability classes), our algorithm was able to find *ideal* strategies where the other approaches consistently failed. We note that, according to the theoretical results proposed in [32], approaches relying on search through the space of uniform strategies may be feasible for models with large information sets. At the same time, they are unlikely to succeed for models with small epistemic classes. This makes our new method a potentially good complement to algorithms like DominoDFS.

*Outline of the Paper* The structure of the paper is as follows. We begin by introducing the standard semantics of strategic ability in Section 2. We also cite the complexity results for model checking and strategy synthesis, and recall the notion of strategic dominance from [35] that will serve as inspiration for our heuristics. In Section 3, we propose an abstract template for multicriterial strategic dominance, and instantiate it by two actual dominance relations that will provide the heuristics. Our algorithm for strategy synthesis based on iterated improvement is presented in Section 4, and evaluated experimentally in Section 5. We also discuss how the algorithm can be extended to synthesis of coalitional strategies in Section 6. Finally, we conclude in Section 7.

*Related Work* A number of frameworks has been aimed at the verification of strategic properties under imperfect information. Regarding the available tools, the state-of-the-art MAS model checker MCMAS [38, 37] combines efficient symbolic representation of state-space using Binary Decision Diagrams (BDDs) with exhaustive iteration over uniform strategies. A similar approach based on exhaustive search through strategy space is presented in [17]. A prototype tool SMC [43] employs bounded unfoldings of transition relation with strategy exploration and calls to MCMAS. Strategy search with optimisation of partial strategies has been further used in [16, 14, 35]. Most relevant to us, the optimisation in [35] was driven by strategic dominance based on the tightness of the outcome set.

Other recent attempts at feasible verification of uniform strategies include [6, 5, 31] that propose methods for reduction of models with incomplete information, based respectively on abstraction, bisimulation, and partial-order equivalences. Another method [33] avoids the brute-force strategy search by using fixpoint approximations of the input formulas. A prototype tool STV implementing the DominoDFS algorithm and the fixpoint approximation was reported in [34].

We note that all the above approaches try to directly synthesize an ideal (i.e., uniform surely winning) strategy for the given goal. In contrast, our new algorithm starts with a flawed strategy (namely, surely winning but not uniform), and attempts to do iterative improvement. As we show, this may well end up in producing an ideal solution in cases where the other methods are inconclusive. No less importantly, our algorithm produces reasonably good strategies even when

an ideal one cannot be found. The only related work in model checking of multi-agent systems, that we are aware of, is [11, 4] where a theoretical framework was proposed for reasoning about strategies that succeed on "sufficiently many" outcome paths.

## 2 Preliminaries

In this section we recall the standard formal framework used for reasoning about strategies in MAS. To this end, *alternating-time temporal logic* **ATL** [2, 3, 45] is often used. We also recall the notion of dominance for partial strategies, that was proposed in [35].

### 2.1 ATL: What Agents Can Achieve

**ATL** [2, 3, 45] generalizes the branching-time temporal logic **CTL** [21] by replacing the path quantifiers $\mathsf{E}, \mathsf{A}$ with *strategic modalities* $\langle\langle A\rangle\rangle$. Formulas of **ATL** allow to express intuitive statements about what agents (or groups of agents) can achieve. For example, $\langle\langle W, E\rangle\rangle \mathrm{F}\, \mathsf{win}_{\mathsf{WE}}$ says that the players West and Eeast in a game of Bridge can jointly win the game. Formally, the syntax of **ATL** is defined by the following grammar:

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \langle\langle A\rangle\rangle \mathrm{X}\phi \mid \langle\langle A\rangle\rangle \mathrm{G}\phi \mid \langle\langle A\rangle\rangle \phi\, \mathrm{U}\, \phi,$$

where $p \in PV$ is an atomic proposition and $A \subseteq \mathbb{A}\mathrm{gt}$ is a group of agents. We read $\langle\langle A\rangle\rangle\gamma$ as *"A can identify and execute a strategy that enforces $\gamma$,"* X as *"in the next state,"* G as *"now and always in the future,"* and U as *"until."*

### 2.2 Models

We interpret **ATL** [3, 45] specifications over a variant of transition systems where transitions are labeled with combinations of actions, one per agent. Moreover, epistemic relations are used to indicate states that look the same to a given agent. Formally, an *imperfect information concurrent game structure*, or simply a *model*, is given by $M = \langle \mathbb{A}\mathrm{gt}, St, PV, V, Act, d, o, \{\sim_a \mid a \in \mathbb{A}\mathrm{gt}\}\rangle$ which includes a nonempty finite set of agents $\mathbb{A}\mathrm{gt} = \{1, \ldots, k\}$, a nonempty set of states $St$, a set of atomic propositions $PV$ and their valuation $V\colon PV \to 2^{St}$, 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 will 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 $a$. By $[q]_a$ we mean the set of states indistinguishable to agent $a$ from the state $q$. The model is assumed to be *uniform*, in the sense that $q \sim_a q'$ implies $d_a(q) = d_a(q')$. Note that perfect information can be modeled by assuming each $\sim_a$ to be the identity relation.

### 2.3 Strategies

A strategy of an agent $a \in \mathbb{A}\mathrm{gt}$ is a conditional plan that specifies what $a$ is going to do in every possible situation. The details of the definition depend on the observational capabilities of the agent and its memory. In this paper we consider the case of *imperfect information imperfect recall* strategies (sometimes also called *uniform memoryless strategies*), where an agent can observe only a part of the environment (i.e., perceives some states as indistinguishable) and performs the same action every time a given state is reached.

Formally, a uniform strategy for $a$ is a function $\boldsymbol{\sigma}_a\colon St \to Act$ satisfying $\boldsymbol{\sigma}_a(q) \in d_a(q)$ for each $q \in St$ and $\boldsymbol{\sigma}_a(q) = \boldsymbol{\sigma}_a(q')$ for each $q, q' \in St$ such that $q \sim_a q'$. A *collective uniform strategy* $\boldsymbol{\sigma}_A$ for a coalition $A \subseteq \mathbb{A}\mathrm{gt}$ is a tuple of individual strategies, one per agent from $A$.

### 2.4 Outcome Paths

A *path* $\lambda = q_0 q_1 q_2 \ldots$ is an infinite sequence of states such that there is a transition between each $q_i, q_{i+1}$. We use $\lambda[i]$ to denote the $i$th position on path $\lambda$ (starting from $i = 0$) and $\lambda[i, j]$ to denote the part of $\lambda$ between positions $i$ and $j$. Function $out(q, \boldsymbol{\sigma}_a)$ returns the set of all paths that can result from the execution of a strategy $\boldsymbol{\sigma}_a$, beginning at state $q$. Formally:

$out(q, \boldsymbol{\sigma}_a) = \{\lambda = q_0, q_1, q_2 \ldots \mid q_0 = q$ and for each $i = 0, 1, \ldots$ there exists $\langle \alpha^i_{a_1}, \ldots, \alpha^i_{a_k} \rangle$ such that $\alpha^i_a \in d_a(q_i)$ for every $a \in \mathbb{A}\mathrm{gt}$, and $\alpha^i_a = \boldsymbol{\sigma}_A|_a(q_i)$ for every $a \in A$, and $q_{i+1} = o(q_i, \alpha^i_{a_1}, \ldots, \alpha^i_{a_k})\}$.

Moreover, the function $out^{\mathrm{ir}}(q, \boldsymbol{\sigma}_a) = \bigcup_{a \in A} \bigcup_{q \sim_a q'} out(q', \boldsymbol{\sigma}_a)$ collects all the outcome paths that start from states that are indistinguishable from $q$ to at least one agent in $A$.

### 2.5 Semantics of ATL

Given a model $M$ and a state $q$, the semantics of **ATL** formulas is defined as follows:

- $M, q \models p$ iff $q \in V(p)$,
- $M, q \models \neg\phi$ iff $M, q \not\models \phi$,
- $M, q \models \phi \wedge \psi$ iff $M, q \models \phi$ and $M, q \models \psi$,
- $M, q \models \langle\!\langle A \rangle\!\rangle \mathrm{X}\phi$ iff there exists a uniform strategy $\boldsymbol{\sigma}_A$ such that for all $\lambda \in out^{\mathrm{ir}}(q, \boldsymbol{\sigma}_A)$ we have $M, \lambda[1] \models \phi$,
- $M, q \models \langle\!\langle A \rangle\!\rangle \mathrm{G}\phi$ iff there exists a uniform $\boldsymbol{\sigma}_A$ such that for all $\lambda \in out^{\mathrm{ir}}(q, \boldsymbol{\sigma}_A)$ and $i \in \mathbb{N}$ we have $M, \lambda[i] \models \phi$,
- $M, q \models \langle\!\langle A \rangle\!\rangle \psi \,\mathrm{U}\, \phi$ iff there exists a uniform $\boldsymbol{\sigma}_A$ such that for all $\lambda \in out^{\mathrm{ir}}(q, \boldsymbol{\sigma}_A)$ there is $i \in \mathbb{N}$ for which $M, \lambda[i] \models \phi$ and $M, \lambda[j] \models \psi$ for all $0 \leq j < i$.

The standard boolean operators (logical constants $\top$ and $\bot$, disjunction $\vee$, and implication $\rightarrow$) are defined as usual. Additionally, we define *"now or sometime in the future"* as $\mathrm{F}\varphi \equiv \top \mathrm{U}\, \varphi$. It is easy to see that $M, q \models \langle\!\langle A \rangle\!\rangle \mathrm{F}\phi$ iff there exists a collective uniform strategy $\boldsymbol{\sigma}_A$ such that, on each path $\lambda \in out^{\mathrm{ir}}(q, \boldsymbol{\sigma}_A)$, there is a state that satisfies $\phi$.

### 2.6 Model Checking and Strategy Synthesis

It is well known that model checking of **ATL** based on uniform memoryless strategies is $\mathbf{\Delta_2^P}$-complete with respect to the size of the explicit (global) model [45, 30, 10], i.e., on top of the usual state-space and transition-space explosion which arises from the composition of the agents' local models. This concurs with the results for solving imperfect information games and synthesis of winning strategies, which are also known to be hard [20, 26, 41]. Note that model checking **ATL** corresponds very closely to strategy synthesis for reachability/safety games. In fact, most model checking algorithms for **ATL** try to build a winning strategy when checking if such a strategy exists.

It is also known that both strategy synthesis and **ATL** model checking for imperfect information are not only theoretically hard, they are also difficult in practice. In particular, imperfect information strategies do not admit straightforward fixpoint algorithms based on standard short-term ability operators [12, 23]. That makes incremental synthesis of strategies impossible, or at least difficult to achieve. Some practical attempts to overcome the barrier have been reported in [14–16, 28, 42, 33, 35]. Up until now, experimental results confirm that the initial intuition was right: model checking of strategic modalities for imperfect information is hard, and dealing with it requires innovative algorithms and verification techniques.

We emphasize that, at the same time, model checking for perfect information strategies (i.e., ones that can specify different choices at indistinguishable states) is much cheaper computationally, namely **P**-complete in the size of the model [3].

### 2.7 Partial Strategies and Strategy Dominance

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 $\Sigma_A$.

Let $q \in dom(\sigma_A)$ for some $\sigma_A \in \Sigma_A$. The *outcome* of $\sigma_A$ 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:

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

where $|\lambda|$ denotes the length (i.e., the number of states) of $\lambda$ and $\lambda$ is either infinite or cannot be extended. For each $i \in \mathbb{N}$ let $\lambda_i$ denote the $i$–th element. Let $Q \subseteq dom(\sigma_A)$. A partial strategy $\sigma_A$ is *$Q$-loopless*, if the set $\bigcup_{q \in Q} out(q, \sigma_A)$

contains only finite paths. For each $p \in PV$ we say that $\sigma_A$ is $p$-free if $V(p) \cap dom(\sigma_A) = \emptyset$.

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

The paper [35] proposed a notion of strategic dominance defined with respect to a given context. Assume that we want to compare two partial strategies $\sigma_A$ and $\sigma'_A$. First, we fix a context strategy $\sigma^C_A$, such that after executing it the control can be given to strategy $\sigma_A$ or $\sigma'_A$. Then, we say that $\sigma_A$ dominates $\sigma'_A$, iff the sets of input states[4] of both strategies are equal, and the set of output states of strategy $\sigma_A$ is a subset of the set of output states of strategy $\sigma'_A$.

## 3 Two Notions of Dominance for Iterated Strategy Improvement

In [35], partial strategies are optimized according to only one criterion, namely the tightness of their outcome sets. In contrast, we propose to use two dimensions for optimization: tightness of the outcome *and* uniformity of the actions selected within the strategy. This is because, unlike [35], we start the synthesis with a perfect information strategy. Thus, our algorithm optimizes strategies that can include any number of conflicts, in the sense that it might prescribe different actions within the same information set $[q]_a$.

### 3.1 Multi-Criterial Domination: Abstract Template

Consider a set of partial strategies $\Sigma$ of agent $a$, based on the same epistemic class of $a$. That is, there exists $q \in St$ such that $dom(\sigma) \subseteq [q]_a$ for every $\sigma \in \Sigma$. Let $\sigma_1, \sigma_2 \in \Sigma$. We begin with an abstract definition of domination that looks at two criteria $\mathcal{C}_1, \mathcal{C}_2$. The idea is that $\sigma_2$ dominates $\sigma_1$ if it improves on $\mathcal{C}_1$ without deteriorating with respect to $\mathcal{C}_2$.

**Definition 1 ($(\mathcal{C}_1,\mathcal{C}_2)$-domination).** *Let each $\mathcal{C}_i$ be a criterion associated with a total order $\preceq_{\mathcal{C}_i}$ on the partial strategies in $\Sigma$. The strict variant $\prec_{\mathcal{C}_i}$ of the ordering is defined in the obvious way, by $\preceq_{\mathcal{C}_i} \setminus (\preceq_{\mathcal{C}_i})^{-1}$. We say that $\sigma_1$* is $(\mathcal{C}_1,\mathcal{C}_2)$-dominated by $\sigma_2$ *iff it holds that $\sigma_1 \prec_{\mathcal{C}_1} \sigma_2$ and at the same time $\sigma_1 \preceq_{\mathcal{C}_2} \sigma_2$.* □

**Definition 2 (Better and best domination).** *Consider partial strategies $\sigma_2, \sigma'_2$ that both $(\mathcal{C}_1,\mathcal{C}_2)$-dominate $\sigma_1$. We say that $\sigma_2$* better $(\mathcal{C}_1,\mathcal{C}_2)$-dominates $\sigma_1$ *iff $\sigma'_2 \prec_{\mathcal{C}_1} \sigma_2$, i.e., $\sigma_2$ performs better than $\sigma'_2$ with respect to the primary criterion $\mathcal{C}_1$. Note: the fact that $\sigma_2$ better dominates $\sigma_1$ than $\sigma'_2$ does not imply that $\sigma_2$ dominates $\sigma'_2$, because $\sigma_2$ may perform poorer than $\sigma'_2$ on the secondary criterion $\mathcal{C}_2$.*

*Moreover, $\sigma_2$* best dominates $\sigma_1$ *with respect to $(\mathcal{C}_1,\mathcal{C}_2)$ iff it dominates $\sigma_1$ and no other strategy in $\Sigma$ better dominates $\sigma_1$. The set of strategies that best dominate $\sigma_1$ with respect to $(\mathcal{C}_1,\mathcal{C}_2)$ will be denoted by $Best_{\mathcal{C}_1,\mathcal{C}_2}(\sigma_1)$.* □

[4] i.e., initial states of the strategy

### 3.2 Outcome- and Uniformity-Dominance

In the following, we assume a shared set of input nodes $In \subseteq dom(\sigma_1), dom(\sigma_2)$. The set of states reachable from $In$ by partial strategy $\sigma_i$ is denoted by $Reach(In, \sigma_i)$. Furthermore, we define the *domain of relevance* of $\sigma_i$ as $RDom(In, \sigma_i) = dom(\sigma_i) \cap Reach(In, \sigma_i)$. That is, $RDom(In, \sigma_i)$ excludes from the domain of $\sigma_i$ the states that cannot be reached, and hence are irrelevant when reasoning about potential conflicts between choices.

The *outcome criterion* is given by relation $\preceq_{\mathcal{O}(In)}$ such that $\sigma_1 \preceq_{\mathcal{O}(In)} \sigma_2$ iff $Reach(In, \sigma_2) \subseteq Reach(In, \sigma_1)$, i.e., $\sigma_2$ has at least as tight set of reachable outcome states as $\sigma_1$.

We will now proceed to the other criterion, related to uniformity of strategies. First, we define the *conflict set* of $\sigma_i$ on states $Q \subseteq St$ as $Conflicts(Q, \sigma_i) = \{(q, q') \in Q \times Q \mid \sigma_i(q) \neq \sigma_i(q')\}$, i.e., the set of all pairs of states from $Q$ where $\sigma_i$ specifies conflicting choices.
Now, the *uniformity criterion* is given by relation $\preceq_{\mathcal{U}(In)}$ such that $\sigma_1 \preceq_{\mathcal{U}(In)} \sigma_2$ iff $Conflicts(RDom(In, \sigma_2), \sigma_2) \subseteq Conflicts(RDom(In, \sigma_1), \sigma_1)$. In other words, all the conflicts that $\sigma_2$ encounters in its domain of relevance must also appear in $\sigma_1$ (but not necessarily vice versa).

**Definition 3 (Outcome- and uniformity-domination).** *We say that* $\sigma_1$ is outcome-dominated by $\sigma_2$ on input $In$ *iff it is* $(\mathcal{O}(In), \mathcal{U}(In))$*-dominated by* $\sigma_2$. *Likewise,* $\sigma_1$ is uniform-dominated by $\sigma_2$ on input $In$ *iff it is* $(\mathcal{U}(In), \mathcal{O}(In))$*-dominated by* $\sigma_2$. *The concepts of better and best domination apply in a natural way.* □

## 4 Iterated Strategy Synthesis

In this section, we propose an algorithm for strategy synthesis, based on the following idea: first generate a surely winning perfect information strategy (if it exists), and then iteratively improve it with respect to the dominance relations proposed in Section 3. Of the two relations, uniformity-dominance has higher priority. The iterative improvement terminates when the procedure reaches a fixpoint (i.e., no more improvement is possible anymore) or when the time limit is exceeded. After that, the optimized strategy is returned and checked for uniformity.

We will now define our procedure in more detail.

**Definition 4 (Input).** *The input of the algorithm consists of: model* $M$*, state* $q$ *in* $M$*, and formula* $\langle\langle a \rangle\rangle \mathrm{F}\varphi$*. We define the set of* initial states *as* $Q_0 = [q]_{\sim_a}$*, i.e., the states that agent* $a$ *considers possible when the system is in* $q$*.*

**Definition 5 (Data structures).** *The algorithm uses the following data structures:*

- *The model;*

**Algorithm 1** Synthesis algorithm $strat\_synth(M)$

Generate a winning perfect information strategy $\boldsymbol{\sigma}$
**if** $\boldsymbol{\sigma}$ doesn't exist **then**
  return **false**
**end if**
Create an empty list $PStr$
Create a list $\mathcal{IS}$ of information sets in $M \dagger \boldsymbol{\sigma}$
**for** $i = 1$ **to** $|\mathcal{IS}|$ **do**
  Take the info set $(i, Q_i)$ and generate the corresponding partial strategy $\sigma_i$ as a restriction of $\boldsymbol{\sigma}$ to $Q_i$ and add it to $PStr$
  $In_i := Q_i \cap Reach\big(\, Reach(Q_0, \boldsymbol{\sigma}) \setminus Q_i,\ (\boldsymbol{\sigma} \setminus \sigma_i)\, \big)$
  $RDom_i := Q_i \cap Reach(In_i, \sigma_i)$
  $Out_i := Reach(In_i, \sigma_i) \setminus Q_i$
  $Conflicts_i := Conflicts(RDom_i, \sigma_i)$
**end for**
Optimize the resulting list of partial strategies $PStr$
**return** $PStr$

- *A list of* information sets *for agent $a$, represented by pairs $(id, Q_{id})$ where $id \in \mathbb{N}$ is the identifier of the info set, and $Q_{id} \subseteq St$ is an abstraction class of the $\sim_a$ relation;*
- *A list of* partial strategies *$PStr$ represented by the following tuples:*
$$(id, \sigma_{id}, In_{id}, RDom_{id}, Conflicts_{id}, Out_{id})$$
*where id is the identifier of the information set on which the strategy operates, $\sigma_{id}$ is the current set of choices, $In_{id}$ the set of input states, $RDom_{id}$ is the domain of relevance of $\sigma_{id}$ from $In_{id}$, $Conflicts_{id}$ is the current set of conflicts, and $Out_{id}$ is the set of output states, i.e., the states by which $\sigma_{id}$ can pass the control to another partial strategy.*

The main part of the procedure is defined by Algorithms 1, 2 and 3. Algorithm 1 tries to generate a perfect information strategy by employing a standard algorithm, e.g., the well-known fixpoint algorithm of [3]. If successful, it produces:

- An ordered list of epistemic indistinguishability classes, also known as *information sets*, for agent $a$. The list is generated by means of depth-first search through the transition network, starting from the initial state. Note that the information sets are restricted to the pruning of model $M$ by strategy $\sigma$, denoted $M \dagger \boldsymbol{\sigma}$ in the pseudocode. That is, only states reachable by $\sigma$ from the initial state will be taken into account when looking at potential conflicts between $a$'s choices;
- The ordered list of partial strategies extracted from $\sigma$, following the same ordering that was established for the information sets.

After that, Algorithm 1 calls Algorithm 3.

Algorithm 3 proceeds in cycles. In each cycle it calls Algorithm 2, which optimizes the partial strategies one by one, following the ordering established by

**Algorithm 2** Single sweep optimization algorithm *optimize_once*($PStr$)

$OldPStr := PStr$
**for** $i = 1$ **to** $|\mathcal{IS}|$ **do**
  **repeat**
    $OldPStr_i := PStr(i)$
    **if** exists $\sigma$ that uniform-best dominates $PStr(i)$ in $In_i$ **then**
      update $PStr(i)$ by taking $\sigma_i := \sigma$ and recomputing the sets $RDom_i$, $Out_i$, and $Conflicts_i$
    **end if**
    **if** exists $\sigma$ that outcome-best dominates $PStr(i)$ in $In_i$ **then**
      update $PStr(i)$ by taking $\sigma_i := \sigma$ and recomputing the sets $RDom_i$, $Out_i$, and $Conflicts_i$
    **end if**
  **until** $PStr(i) = OldPStr_i$
  update $\boldsymbol{\sigma}$ with the current contents of $PStr$
  **for** every $j \neq i$ **do**
    update the input states of $PStr(j)$ by $In_j := Q_j \cap Reach\big(\, Reach(Q_0, \boldsymbol{\sigma}) \setminus Q_j,\ (\boldsymbol{\sigma} \setminus \sigma_j)\,\big)$
  **end for**
**end for**
**return** $PStr$

**Algorithm 3** Optimization algorithm *optimize*($PStr$)

**repeat**
  $OldPStr := PStr$
  $Pstr := optimize\_once(PStr)$
**until** timeout or (PStr = OldPStr)
**return** $PStr$

Algorithm 1. Moreover, each partial strategy is optimized first with respect to the uniformity-dominance, and then according to the outcome-dominance; this proceeds in a loop until a fixpoint is found. Algorithm 3 terminates when no improvement has been seen in the latest iteration, or the timeout is reached.

It is worth emphasizing that, except for the first phase (generation of a perfect information strategy), this is an anytime algorithm. It means that the procedure will return *some* strategy even for models whose size is beyond grasp for optimal model checking algorithms. This is a clear advantage over the existing approaches [38, 37, 16, 14, 43, 33, 35] where the algorithms typically provide no output even for relatively small models.

## 5 Experimental Evaluation

We evaluate the algorithm of Section 4 through experiments with two classes of models: randomly generated models and the Drones benchmark of [35].

### 5.1 First Benchmark: Random Models

As the first benchmark for our experiments, we use randomly generated models of a given size. The models represent a single agent playing against a nondeterministic environment. The models are generated according to the following procedure. First, we begin by generating a directed graph with several, randomly chosen, connections. The size of the graph is given by the parameter. Subsequently, we introduce additional connections between randomly selected nodes from distinct paths, in order to increase the complexity of the resulting model. Winning states are selected from the set containing the final states from each of the paths.

Once the graph is generated, it is used to construct the model. Each node represents a unique state, and a connection between two nodes indicates the presence of at least one transition between them. The transitions are generated using the following approach: for each node, a subset of outgoing connections is randomly chosen. From this subset, a set of transitions is created with actions selected randomly. As a result, some transitions will be influenced not only by the agent but also by the nondeterministic environment. This process is repeated multiple times. In the final step of the model generation algorithm, atomic propositions are randomly assigned to states, and epistemic classes are generated at random.

The number of connections, actions, winning states and epistemic classes is given as the function of the number of states in the model.

### 5.2 Second Benchmark: Drone Model

As the second benchmark we use the Drone Model from [35] with some minor modifications. In this scenario drones are used to measure the air quality in the specified area. The motivation is clear, as nowadays many cities face a problem of air pollution.

A model is described using three variables:

- Number of drones;
- Initial energy for each drone;
- Map size, i.e., the number of places in the area.

Every drone is equipped with a limited battery, initially charged to some energy level. Each action that the drone performs uses one energy unit. When the battery is depleted, the drone lands on the ground and must be picked up.

In our scenario, in contrast to the original one, the map is randomly generated as a directed graph. This introduces randomization into the model generation process, enabling us to thoroughly test our algorithms. It is guaranteed that the graph is connected, and each node can be reached from the initial one. Furthermore, each node has no more than four neighbors: one for each direction of the world. Along with the map, pollution readings are also randomly generated and assigned to each place. Readings can have one of the two values: pollution or no pollution.

| | | Strategy Perfect Info | | | Simplified Strategy | | | | Approximation | | Domino DFS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| #st | G. time | G. time | #str | #ep | G. time | #str | #ep | %ir | Time | Conclusive | Time | True |
| 10 | 0.014 | 0.031 | 10 | 5 | 42.033 | 7 | 0 | 100% | 0.018 | 50% | 0.57 | 100% |
| 100 | 0.176 | 0.546 | 92 | 61 | 60.210 | 83 | 0 | 100% | 0.519 | 20% | 90 | TIMEOUT |
| 1000 | 9.401 | 22.001 | 882 | 629 | 61.865 | 780 | 0 | 100% | 3.136 | 0% | 90 | TIMEOUT |

**Fig. 1.** Random Model results with logarithmic epistemic classes

| | | Strategy Perfect Info | | | Simplified Strategy | | | | Approximation | | Domino DFS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| #st | G. time | G. time | #str | #ep | G. time | #str | #ep | %ir | Time | Conclusive | Time | True |
| 10 | 0.009 | 0.023 | 10 | 5 | 24.048 | 6 | 3 | 20% | 0.017 | 80% | 1.14 | 100% |
| 100 | 0.202 | 0.489 | 94 | 58 | 54.253 | 66 | 36 | 10% | 0.197 | 0% | 90 | TIMEOUT |
| 1000 | 10.817 | 25.239 | 917 | 584 | 61.496 | 614 | 347 | 10% | 2.647 | 0% | 90 | TIMEOUT |

**Fig. 2.** Random Model results with linear epistemic classes

Each drone holds information about its current energy level, the set of already visited places and its current position on the map. When in a coalition, the drones share their data between themselves, as it is often done in real-life applications. The indistinguishability relations are given by a faulty GPS mechanism: some of the places on the map are indistinguishable for the drone. In that way, epistemic classes are defined.

At each step, the drone can perform one of the listed actions:

- Fly in one of four directions: North, West, South or East;
- Wait, i.e., stay in the current place.

As mentioned before, each action costs the drone one unit of its energy level. Due to the unpredictable nature of the wind, when performing the *fly* action the drone can be carried away to a different place from the one it intended.

### 5.3 Running the Experiments

In the experiments, we have tested 10 cases for each benchmark and each configuration, and collected the average results. Due to the randomized nature of the models, it was possible that the model generation produces a structure where no winning perfect information strategy existed. Such models were disregarded in the output of the experiments. We note in passing that, for the Randomized Model benchmark, winning perfect information strategies existed in approximately 70% of cases.

For each test case, first the perfect information strategy was randomly chosen, and then its optimized version was generated according to Algorithm 3. We compared our results with two other methods: fixpoint approximation from [33] and DominoDFS from [35]. Both algorithms were implemented in Python as well as the strategy optimization algorithm. The code is available online at https://github.com/blackbat13/stv.

Random Model was tested in two different configurations that differ only by the function that binds the size of epistemic classes. In the first configuration,

| | | | Strategy Perfect Info | | | Simplified Strategy | | | | Approximation | | Domino DFS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Map | #st | G. time | G. time | #str | #ep | G. time | #str | #ep | %ir | Time | Conclusive | Time | True |
| 5 | 330 | 0.078 | 0.043 | 38 | 13 | 36.003 | 13 | 1 | 60% | 0.036 | 0% | 9.012 | 90% |
| 10 | 10648 | 3.420 | 1.284 | 74 | 33 | 42.478 | 30 | 5 | 60% | 1.895 | 0% | 90 | TIMEOUT |

**Fig. 3.** Drone Model results

the maximum size of the epistemic classes was given by $\log_2 n$, where $n$ is the the number of states in the model. In the second configuration, the size of the epistemic classes was at most 10% $n$, i.e., linear wrt to the size of the state space.

For both benchmarks, only singleton coalitions were considered. In particular, for Drone Model, we only generated models with a single drone acting against the environment.[5] The initial energy of the drone was defined as the number of places in the map times two, in order to increase the likelihood of generating a model in which the drone can visit all the places on the randomly generated map.

The experiments were conducted on an Intel Core i7-6700 CPU with dynamic clock speed of 2.60–3.50 GHz, 32 GB RAM, running under 64bit Linux Debian.

### 5.4 Results

The output of the experiments is presented in Figures 1, 2 and 3. Figures 1 and 2 present the results for the Random Model benchmark; Figure 3 presents the results for the Drone Model benchmark. All running times are given in seconds. The timeout was set to 90 seconds. In case of strategy optimization, this was split into two parts: 30 seconds for the strategy generation, and 60 second for its optimization.

The first columns present information about the model configuration, its size and generation time. The next seven columns describe the output of our algorithms, i.e., the randomly generated strategy with perfect information and its optimized version. The last part of the tables contains the reference results from the algorithms used for comparison: lower and upper fixpoint approximation and DominoDFS method.

The table headers should be interpreted as follows:

- *Map*: number of places on the map (for Drone Model);
- *#st*: number of states in the model;
- *G.time*: generation time for the model/strategy;
- *#str*: number of states reachable in the strategy;
- *#ep*: number of states in which the strategy uniformity was broken;
- *%ir*: percentage of cases in which optimized strategy was a uniform strategy;
- *Time*: time used by the Approximation/Domino DFS algorithm;
- *Conclusive*: percentage of cases in which the result of fixpoint approximation was conclusive, i.e. when both the upper bound and the lower bound computations yield the same outcome;

[5] Preliminary experiments for coalitions of drones are presented in Section 6.

**Algorithm 4** Optimization algorithm for coalition $optimize\_coal(PStr, A)$

```
repeat
  OldPStr := PStr
  for agent in A do
    Pstr := optimize_once(PStr)
  end for
until timeout or (PStr = OldPStr)
return PStr
```

– $True$: percentage of cases in which Domino DFS returned a winning strategy (timeout was reached in all the other cases).

As the results show, our method performed very well in comparison to the reference algorithms. The DominoDFS method ended mostly with timeout for larger models, and the fixpoint approximations gave mostly inconclusive results. In contrast, our optimized strategies obtained pretty good elimination of conflicts, and in many cases produced ideal, i.e., fully uniform strategies.

The results also show clearly that our optimization algorithm works best in situations when the size of the epistemic classes is relatively small. For the logarithmic size of the epistemic classes, the optimized strategy was always a uniform strategy (!). As for the setting with the linear size, the optimization-based algorithm was not as good, but still gave a reduction of conflicts of about 40%. Even in that case, it produced ideal strategies in $10 - 20\%$ of instances. It is also worth pointing out that, for the Drone benchmark, our optimization returned a uniform strategy in about 60% cases.

We note, again, that our algorithm is an anytime algorithm, which means that it always returns *some* strategy, regardless of the given timeout.

## 6 Coalitional Strategies

So far, we have focused on the synthesis of individual strategies. In fact, our synthesis algorithm in Section 4 works only for singleton coalitions. This is because it relies on the fact that the domains of partial strategies are closed with respect to indistinguishability relations of the involved agents. While such a closure is guaranteed for information sets of single agent, the union of information sets of several agents typically does not satisfy the property.

One way out is to *define* the domains of partial strategies by the closure. The domains would in that case correspond to common knowledge neighborhoods for the coalition. Unfortunately, this will not work well in practice: for most models, the common knowledge closure will produce the whole state space, and thus make the computation infeasible.

Another simple idea is to optimize coalitional strategies agent-wise, alternating between the agents. In that case, we optimize the individual strategies being parts of $\sigma_A$ one by one, using the optimization template from Section 4. The resulting procedure is presented as Algorithm 4.

| | | | Strategy Perfect Info | | | Simplified Strategy | | | |
|---|---|---|---|---|---|---|---|---|---|
| Map | #st | G. time | G. time | #st | #ep | G. time | #st | #ep | %ir |
| 3 | 667 | 0.85 | 0.397 | 35 | 11 | 12.031 | 9 | 1 | 60% |
| 5 | 31122 | 69.265 | 107.428 | 587 | 728 | 60.8 | 87 | 58 | 40% |

**Fig. 4.** Drone model results for coalitions

The output of our experimental evaluation for synthesis of coalitional strategies is presented in Figure 4. For the experiments, the Drone benchmark was selected with coalition of two drone agents. As the results show, our algorithm obtained a high level of optimization of the initial, perfect information, strategy. Most importantly, the procedure produced ideal strategies in 60% and 40% of the instances, respectively, thus providing a conclusive answer to the model checking question in about half of the cases.

## 7 Conclusions

In this paper, we propose an anytime algorithm to synthesize "reasonably good" strategies for reachability goals under imperfect information. The idea is to first generate a surely winning strategy with *perfect information*, and then iteratively improve it with respect to its uniformity level and the tightness of its outcome set. We evaluate the algorithm experimentally on two classes of models: randomly generated ones and ones modeling a group of drones patrolling for air pollution. The results show high optimization rates, especially for models with relatively small indistinguishability classes. For such models, the procedure produced ideal strategies in a large fraction of the instances, thus providing a conclusive answer to the model checking question.

The fact that our method works well for models with small epistemic classes suggests that it should complement, rather than compete, with methods based on search through the space of uniform strategies (which usually work better for models with *large* information sets). Depending on the kind of the model, a suitable algorithm should be used.

**Acknowledgements** The work was supported by NCBR Poland and FNR Luxembourg under the PolLux/FNR-CORE projects STV (POLLUX-VII/1/2019 and C18/IS/12685695/IS/STV/Ryan), SpaceVote (POLLUX-XI/14/SpaceVote/2023 and C22/IS/17232062/SpaceVote) and PABLO (C21/IS/16326754/PABLO). The work of Damian Kurpiewski was also supported by the CNRS IEA project MoSART.

## References


1. Alur, R., de Alfaro, L., Henzinger, T.A., Krishnan, S., Mang, F., Qadeer, S., Rajamani, S., Tasiran, S.: MOCHA: Modularity in model checking. Tech. rep., University of Berkeley (2000)
2. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time Temporal Logic. In: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS). pp. 100–109. IEEE Computer Society Press (1997)
3. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time Temporal Logic. Journal of the ACM **49**, 672–713 (2002). https://doi.org/10.1145/585265.585270
4. Aminof, B., Malvone, V., Murano, A., Rubin, S.: Graded modalities in Strategy Logic. Information and Computation **261**, 634–649 (2018). https://doi.org/10.1016/j.ic.2018.02.022
5. Belardinelli, F., Condurache, R., Dima, C., Jamroga, W., Jones, A.: 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). pp. 1286–1295. IFAAMAS (2017)
6. Belardinelli, F., Lomuscio, A.: Agent-based abstractions for verifying alternating-time temporal logic with imperfect information. In: Proceedings of AAMAS. pp. 1259–1267. ACM (2017)
7. Berthon, R., Maubert, B., Murano, A., Rubin, S., Vardi, M.Y.: Strategy logic with imperfect information. In: Proceedings of LICS. pp. 1–12 (2017). https://doi.org/10.1109/LICS.2017.8005136
8. Berthon, R., Maubert, B., Murano, A., Rubin, S., Vardi, M.Y.: Strategy logic with imperfect information. ACM Trans. Comput. Log. **22**(1), 5:1–5:51 (2021). https://doi.org/10.1145/3427955, https://doi.org/10.1145/3427955
9. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Trans. on Computers **35**(8), 677–691 (1986)
10. Bulling, N., Dix, J., Jamroga, W.: Model checking logics of strategic ability: Complexity. In: Dastani, M., Hindriks, K., Meyer, J.J. (eds.) Specification and Verification of Multi-Agent Systems, pp. 125–159. Springer (2010)
11. Bulling, N., Jamroga, W.: What agents can probably enforce. Fundamenta Informaticae **93**(1-3), 81–96 (2009)
12. Bulling, N., Jamroga, W.: Alternating epistemic mu-calculus. In: Proceedings of IJCAI-11. pp. 109–114 (2011)
13. Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L., Hwang, L.J.: Symbolic model checking: 10^20 states and beyond. In: Proc. of 4th Ann. IEEE Symp. on Logic in Computer Science (LICS). pp. 428–439. IEEE Computer Society (1990)
14. Busard, S.: Symbolic Model Checking of Multi-Modal Logics: Uniform Strategies and Rich Explanations. Ph.D. thesis, Universite Catholique de Louvain (2017)
15. Busard, S., Pecheur, C., Qu, H., Raimondi, F.: Improving the model checking of strategies under partial observability and fairness constraints. In: Formal Methods and Software Engineering, Lecture Notes in Computer Science, vol. 8829, pp. 27–42. Springer (2014). https://doi.org/10.1007/978-3-319-11737-9_3
16. Busard, S., Pecheur, C., Qu, H., Raimondi, F.: Reasoning about memoryless strategies under partial observability and unconditional fairness constraints. Information and Computation **242**, 128–156 (2015). https://doi.org/10.1016/j.ic.2015.03.014
17. Calta, J., Shkatov, D., Schlingloff, B.H.: Finding uniform strategies for multi-agent systems. In: Proceedings of Computational Logic in Multi-Agent Systems (CLIMA). Lecture Notes in Computer Science, vol. 6245, pp. 135–152. Springer (2010)

18. Cermak, P., Lomuscio, A., Mogavero, F., Murano, A.: 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, pp. 525–532. Springer (2014)
19. Cermák, P., Lomuscio, A., Murano, A.: Verifying and synthesising multi-agent systems against one-goal strategy logic specifications. In: Proceedings of AAAI. pp. 2038–2044 (2015)
20. Chatterjee, K., Doyen, L., Henzinger, T., Raskin, J.F.: Algorithms for omega-regular games of incomplete information. Logical Methods in Computer Science **3**(3) (2007)
21. Clarke, E., Emerson, E.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Proceedings of Logics of Programs Workshop. Lecture Notes in Computer Science, vol. 131, pp. 52–71 (1981)
22. Dembiński, P., Janowska, A., Janowski, P., Penczek, W., Półrola, A., Szreter, M., Woźna, B., Zbrzezny, A.: 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, pp. 278–283. Springer (2003)
23. Dima, C., Maubert, B., Pinchinat, S.: The expressive power of epistemic $\mu$-calculus. CoRR **abs/1407.5166** (2014)
24. Dima, C., Maubert, B., Pinchinat, S.: 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, pp. 179–191. Springer (2015). https://doi.org/10.1007/978-3-662-48057-1_14
25. Dima, C., Tiplea, F.: Model-checking ATL under imperfect information and perfect recall semantics is undecidable. CoRR **abs/1102.4225** (2011)
26. Doyen, L., Raskin, J.F.: Games with imperfect information: Theory and algorithms. In: Lecture Notes in Game Theory for Computer Scientists, pp. 185–212. Cambridge University Press (2011)
27. Gammie, P., Meyden, R.: MCK: Model checking the logic of knowledge. In: Proc. of the 16th Int. Conf. on Computer Aided Verification (CAV'04). LNCS, vol. 3114, pp. 479–483. Springer-Verlag (2004)
28. Huang, X., van der Meyden, R.: Symbolic model checking epistemic strategy logic. In: Proceedings of AAAI Conference on Artificial Intelligence. pp. 1426–1432 (2014)
29. Jamroga, W.: Logical Methods for Specification and Verification of Multi-Agent Systems. ICS PAS Publishing House (2015)
30. Jamroga, W., Dix, J.: Model checking $\mathrm{ATL}_{ir}$ is indeed $\Delta_2^P$-complete. In: Proceedings of EUMAS. CEUR Workshop Proceedings, vol. 223 (2006)
31. Jamroga, W., Penczek, W., Dembiński, P., Mazurkiewicz, A.: Towards partial order reductions for strategic ability. In: Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 156–165. IFAAMAS (2018)
32. Jamroga, W., Knapik, M.: Some things are easier for the dumb and the bright ones (beware the average!). In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence IJCAI. pp. 1734–1740 (2019). https://doi.org/10.24963/ijcai.2019/240
33. Jamroga, W., Knapik, M., Kurpiewski, D., Mikulski, Ł.: Approximate verification of strategic abilities under imperfect information. Artificial Intelligence **277** (2019). https://doi.org/10.1016/j.artint.2019.103172

34. Kurpiewski, D., Jamroga, W., Knapik, M.: STV: Model checking for strategies under imperfect information. In: Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2019. pp. 2372–2374. IFAAMAS (2019)
35. Kurpiewski, D., Knapik, M., Jamroga, W.: On domination and control in strategic ability. In: Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2019. pp. 197–205. IFAAMAS (2019)
36. Kurpiewski, D., Pazderski, W., Jamroga, W., Kim, Y.: STV+Reductions: Towards practical verification of strategic ability using model reductions. In: Proceedings of AAMAS. pp. 1770–1772. ACM (2021)
37. Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: An open-source model checker for the verification of multi-agent systems. International Journal on Software Tools for Technology Transfer **19**(1), 9–30 (2017). https://doi.org/10.1007/s10009-015-0378-x
38. Lomuscio, A., Raimondi, F.: Model checking knowledge, strategies, and games in multi-agent systems. In: Proceedings of International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 161–168 (2006). https://doi.org/10.1145/1160633.1160660
39. Mogavero, F., Murano, A., Perelli, G., Vardi, M.: Reasoning about strategies: On the model-checking problem. ACM Transactions on Computational Logic **15**(4), 1–42 (2014)
40. Mogavero, F., Murano, A., Vardi, M.: Reasoning about strategies. In: Proceedings of FSTTCS. pp. 133–144 (2010)
41. Peterson, G., Reif, J.: Multiple-person alternation. In: Proceedings of the 20th Annual Symposium on Foundations of Computer Science (FOCS). pp. 348–363. IEEE Computer Society Press (1979)
42. Pilecki, J., Bednarczyk, M., Jamroga, W.: Synthesis and verification of uniform strategies for multi-agent systems. In: Proceedings of CLIMA XV. Lecture Notes in Computer Science, vol. 8624, pp. 166–182. Springer (2014)
43. Pilecki, J., Bednarczyk, M., Jamroga, W.: SMC: Synthesis of uniform strategies and verification of strategic ability for multi-agent systems. Journal of Logic and Computation **27**(7), 1871–1895 (2017). https://doi.org/10.1093/logcom/exw032
44. Raimondi, F., Lomuscio, A.: Automatic verification of multi-agent systems by model checking via ordered binary decision diagrams. J. Applied Logic **5**(2), 235–251 (2007)
45. Schobbens, P.Y.: Alternating-time logic with imperfect recall. Electronic Notes in Theoretical Computer Science **85**(2), 82–93 (2004)